VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Návrh architektury a vývoj softwarových komponent pro integraci informačních systémů ve firmě VRK plus s.r.o.

Design of Architecture and Development of Software Components for Integration of Information Systems in VRK plus s.r.o. Company

2016 Bc. Martin Gelnár

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Gelnár**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Návrh architektury a vývoj softwarových komponent pro integraci informačních systémů ve firmě VRK plus s.r.o.
Design of Architecture and Development of Software Components for Integration of Information Systems in VRK plus s.r.o. Company

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je analýza architektury současných systémů firmy, implementace komponent pro integraci a návrh procesu nasazení nových verzí u zákazníků v reálném provozu.

Jednotlivé body zadání:
1. Analýza používaných vývojových nástrojů a frameworků ve firmě.
2. Návrh optimalizovaných procesních postupů vývoje a používaných vývojových nástrojů.
3. Analýza stávajících produktů a identifikace integrovatelných částí.
4. Návrh procesu integrace vývoje produktů.
5. Implementace klíčových komponent určených k integraci.
6. Návrh postupu nasazení optimalizovaných produktů u zákazníka.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

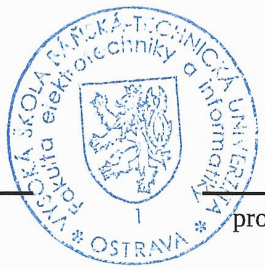
Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Robert Kubáček**

Konzultant diplomové práce: doc. Mgr. Miloš Kudělka, Ph.D.

Datum zadání: 01.09.2014

Datum odevzdání: 29.04.2016



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2016

.....

VRK plus s.r.o.
Fr. Lýska 1605/3
700 30 Ostrava-Bělský Les
IČ: 60321580, DIČ: CZ60321580

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

V dnešní době rychlého rozvoje techniky musí firmy vyvíjející software držet krok s posledními trendy. Software postavený na zastaralých technolgiích a frameworcích se obtížně udržuje a inovuje. Proto firma musí volit takové technologie, které budou v budoucnu stále udržované a dostupné. Tato práce popisuje současný stav procesu vývoje softwarových produktů ve firmě VRK plus s.r.o. a navrhuje jeho optimalizaci na základě agilní metody Fearure-Driven Development. Součástí práce je také návrh způsobu sloučení společných funkčností do základních modulů. Pro stávající produkty, které nelze převést nebo budou převedeny později, byl navržen způsob propojení pomocí zasílání zpráv. Práce také definuje časový harmonogram vývoje a nasazení nových produktů u zákazníků.

Klíčová slova: Feature-Driven Development, Enterprise application integration, Zasílání zpráv, Liferay

Abstract

Software developing companies has to follow latest trends in the current era of rapid development of information technologies. Software built on outdated technologies and frameworks are difficult to maintain and innovate. Therefore, companies has to choose technologies which will be still accessible and supported in the future. This thesis describes the current state of the process of software development in VRK plus s.r.o. company and suggests its optimization based on agile Feature-Driven Development methods. This work also includes a proposal on how to merge common functionalities into the core modules. Communication methods based on messaging were designed for existing products which can not be transferred or which will be transferred later. The thesis also defines a schedule for a development and deployment of new products for customers.

Key Words: Feature-Driven Development, Enterprise application integration, Messaging, Liferay

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
BPMN	–	Business Process Model and Notation
CRM	–	Customer Relationship Management
CRUD	–	Create, Read, Update, Delete
CVS	–	Concurrent Version System
FDD	–	Feature-Driven Development
FOP	–	Formatting Objects Processor
FTL	–	FreeMarker Template
GPS	–	Global Positioning System
GWT	–	Google Web Toolkit
HSQLDB	–	HyperSQL DataBase
HTML	–	Hyper Text Markup Language
HTTP	–	Hypertext Transfer Protocol
IDE	–	Integrated Development Environment
JSF	–	JavaServer Faces
JSON	–	JavaScript Object Notation
JSP	–	JavaServer Pages
ORM	–	Object-Relational Mapping
PHP	–	Hypertext Preprocessor
POM	–	Project Object Model
RPC	–	Remote Procedure Call
RUP	–	Rational Unified Process
SWT	–	Standard Widget Toolkit
UUID	–	Universally Unique Identifier

Seznam obrázků

1 Úvod

V portfoliu firmy VRK plus s.r.o. (firma) se nachází několik informačních systémů a aplikací (aplikace), které firma nabízí svým zákazníkům nebo je využívá pro své vnitřní potřeby. Tyto aplikace se začaly vyvíjet postupně v různých časových etapách fungování firmy. Na každou aplikaci byly kladeny různé požadavky na vývoj i provoz. Postupem času některé aplikace využívaly podobných funkčností, které se začaly částečně slučovat do společných modulů.

Snahou firmy v rámci inovace je převést aplikace na jednotnou platformu tak, aby se redukovalo roztříštění použitých technologií a frameworků. Klade se důraz na dostupnost přes internet a funkčnost i na mobilních zařízeních. Požadavkem je členění projektů do modulů obsahující různou funkčnost tak, aby se aplikace daly sestavit a nastavit podle požadavků firmy a jejich zákazníků.

Cílem práce je analýza architektury a stavu současných systémů, aby se zjistilo, které prvky jsou podobné a které by se daly sjednotit. Bude potřeba formalizovat vývojové postupy používané ve firmě a stanovit jednotné vývojové prostředí.

V kapitole 2 se seznámíme s používanými vývojovými nástroji a frameworky ve firmě a s aplikacemi, kterých se bude týkat sjednocení do jednotného prostředí. Kapitola 3 se zaměřuje na optimalizaci procesních postupů vývoje ve firmě pomocí agilní metody Feature-Driven Development. Součástí této kapitoly je i popis vývojových nástrojů, které se budou ve firmě primárně používat při dalším vývoji produktů. Kapitola 4 popisuje podobné části současných produktů, které se budou integrovat do nového prostředí. Kapitola 5 popisuje způsoby a návrh integrace se stávajícími produkty. Obsahuje i návrh procesu integrace vývoje produktů, tzn. jakým způsobem se budou podobné části vyvíjet. Součástí je i návrh a analýza nových společných částí. V kapitole 6 si ukážeme vývoj aplikací pro portál Liferay a ukázku některých implementovaných prvků. Kapitola 7 obsahuje jednotlivé etapy vývoje vedoucí ke sjednocení současných aplikací.

Součástí této práce bude i neveřejná část, která bude obsahovat v elektronické příloze zdrojové kódy implementovaných prvků, které jsou popsané v kapitole 6.1 a 6.2.

2 Analýza používaných vývojových nástrojů a frameworků

Primárním programovacím jazykem používaným ve firmě je Java[1], která se využívá ve většině vytvářených aplikací. Ekosystém Javy nabízí velké množství frameworků a knihoven, které umožňují snadnější a rychlejší řešení různých požadavků a úloh. Její výhodou je i multiplatformnost, která umožňuje, aby firma mohla svým zákazníkům nabídnout aplikace na různých operačních systémech formou instalace pro Microsoft Windows nebo Linux, anebo dostupnost přes webový prohlížeč.

Jako vývojové prostředí se používá Eclipse IDE[2]. Část používaných frameworků nabízí zásuvné moduly pro Eclipse, které usnadňují vývoj aplikací nad těmito frameworky přímo v prostředí Eclipse. Kvůli kompatibilitě starších frameworků se některé aplikace vyvíjí ve starší verzi Eclipse Indigo (verze 3.7) a některé v poslední verzi Eclipse Mars (verze 4.5). Aktualizace starších frameworků do poslední verze Eclipse Mars by byla komplikovaná a časově náročná.

Aplikace využívají pro ukládání dat databázové servery MySQL[3] a PostgreSQL[4], nebo souborový databázový systém HSQLDB[5]. K propojení firemních aplikací s databází se používá framework Hibernate[6], který umožňuje objektově-relační mapování a zajišťuje perzistentci dat.

Některé aplikace jsou naprogramované ve skriptovacím jazyce PHP[7]. Z historického hlediska byly tyto aplikace napsány buď pro interní účely, nebo byly převzaty od jiných vývojářů a začleněny do portfolia firmy. Funkce napsané v PHP nejdou použít spolu s Javou a to komplikuje další vývoj a údržbu. Dále tato skutečnost klade nároky na znalost jazyka PHP u programátorů.

Aplikace vyvíjené firmou se jmenují Smile, ISIS, Klub, Katalog dětských táborů, CRM a portál České-školy.info a jejich popis je uveden níže.

2.1 Smile

Smile je aplikace umožňující školám tvorbu a aktualizaci ŠVP (Školní vzdělávací programy), následně jejich rozpracování do tématických plánů. K dispozici je také možnost tvorby příprav na konkrétní vyučovací hodiny. Pro své uživatele aplikace nabízí interaktivní i statické kontroly při tvorbě ŠVP. Pro firmu aplikace slouží také jako editor RVP (Rámcových vzdělávacích programů), které jsou základem pro ŠVP jednotlivých škol.

Z aplikace lze vygenerovat jednak dílčí výstupy jako jsou například učební plány nebo jednotlivé předměty, tak také komplexní výstupy jako je kompletní ŠVP vytvořené školou. Aplikace dále umožňuje vygenerování interaktivních webových stránek určených pro prezentaci vytvořeného ŠVP na webu.

[1] http://www.java.com/
[2] http://www.eclipse.org/
[3] http://www.mysql.com/
[4] http://www.postgresql.org/
[5] http://www.hsqldb.org/
[6] http://www.hibernate.org/
[7] http://www.php.net/

Aplikace je vyvíjena v programovacím jazyce Java, využívá prostředí Eclipse Indigo, knihovnu grafických uživatelských prvků SWT[8] a JFace[9] (nadstavba nad SWT). Aplikace Smile běží jako desktopová aplikace. Na začátku prodeje se aplikace nabízela s jednouživatelskou licencí a pro ukládání dat se používal souborový databázový systém HSQLDB na lokálním počítači. Později byla do aplikace přidána správa uživatelů a přidána víceuživatelská licence, aby za práci na ŠVP byli zodpovědní konkrétní uživatelé. Nakonec vznikla síťová licence, která umožňuje připojování aplikace k databázovému serveru MySQL umístěném v síti školy, aby mohlo pracovat více uživatelů současně z různých počítačů.

2.2 ISIS

ISIS (Internetový školní informační systém) je určen pro vedení vysokých škol a pomáhá školám s jejich agendou jako správa studentů od podání přihlášky až po ukončení studia, organizaci studia, organizaci práce kantorů, elektronický index studentů, evidence plateb za studium a jiné činnosti spojené s prací na vysoké škole. Systém je provozován na několika vysokých školách a v systému mají uložené informace za několik desítek let.

Systém také umožňuje vygenerovat dokumenty související se studiem (přihláška, diplom, diploma supplement, aj.), generování statistik a exportu dat na ÚIV (Ústav pro informace ve vzdělávání) a SIMS (Sdružené informace matrik studentů), a další.

Systém je vyvíjen v programovacím jazyce Java využívající platformu Enterprise Edition. Vývojovým prostředím je Eclipse Indigo. Jedná se o webovou aplikaci a běží na aplikačním serveru Apache Tomcat[10] umístěného u zákazníka. V průběhu vývoje byly použity postupně dva frameworky Apache Struts[11] 1 a Apache Struts 2. Prezentační vrstva je napsaná v technologii JSP a Apache Tiles[12]. Pro generování dokumentů do PDF se používá Apache FOP[13]. Podle různých požadavků zákazníků jsou data uložená buď v MySQL nebo PostgreSQL.

2.3 Klub

Informační systém Klub slouží pro podporu organizací, které se věnují provozování různých zájmových činností. Typickým zákazníkem jsou spolky (dříve občanská sdružení), příspěvkové organizace, ale i jiné právní formy. Hlavní funkcí systému je evidence členské základny a na ni navazujících činností jako evidence kontaktních údajů, evidence plateb, evidence zákonem daných a dalších údajů, které organizace vedou pro svoji potřebu. Dalšími součástmi jsou evidence dokumentů, docházka, komunikace se členy, statistiky, sestavy, kontroly, evidence firem a obchodních partnerů.

[8] https://www.eclipse.org/swt/
[9] https://wiki.eclipse.org/JFace
[10] http://tomcat.apache.org/
[11] https://struts.apache.org/
[12] https://tiles.apache.org/
[13] https://xmlgraphics.apache.org/fop/

Systém je vyvíjen v programovacím jazyce Java využívající platformu Enterprise Edition a využívá prostředí Eclipse Mars. Jedná se o webovou aplikaci a běží na aplikačním serveru Apache Tomcat umístěného na serveru firmy a nabízí se jako hostovaná aplikace. Systém je vyvíjen ve frameworku GWT[14]. Pro prezentační vrstvu bylo použito externí rozšíření Smart GWT[15], které obsahuje mnoho grafických komponent. Data jsou uložena v MySQL.

2.4 Katalog dětských táborů

Katalog dětských táborů (Katalog táborů) obsahuje nabídku dětských táborů a podobných akcí pro děti. Informace o táborech na tento server vkládají přímo jejich provozovatelé a organizátoři. Návštěvník má možnost hledat tábory a akce prostřednictvím strukturovaného katalogu nebo pomocí vyhledávání po zadání různých parametrů. Součástí je i táborová inzerce s nabídkou nebo poptávkou všeho, co se táborů týče.

Katalog táborů jsem vyvíjel při bakalářském studiu na VŠB-TUO spolu s kamarádem jako mimoškolní projekt. Jelikož jsme oba nastoupili později do zaměstnaneckého poměru, další rozvoj a údržba byla problematická a časově náročná. Došlo později k dohodě s firmou na dalším provozu a vývoji systému a katalog se stal součástí portfolia firmy.

Systém byl vyvíjen ve skriptovacím jazyce PHP. Jedná se o webovou aplikaci a běží na webovém serveru Apache HTTP Server[16] na hostingu firmy. Data jsou ukládána do MySQL.

2.5 CRM

Jedná se o interní systém firmy a slouží pro evidenci zákazníků firmy a jejich kontaktů. Dále se v systému spravuje licenční politika firmy vůči zákazníkům. Tento systém také obsahuje komunikaci se zákazníky a statistiky. Do CRM se posílají provozní data z aplikací u zákazníka. V současné době existují dvě verze CRM. Cílem bylo je obě sloučit do novější verze, ale proces slučování se nepodařilo dokončit.

V první verzi systému se evidují zákazníci pouze z řad škol. CRM bylo přizpůsobeno pro produkty Smile a ISIS, které školy používají. Tento systém byl vytvořen ve skriptovacím jazyce PHP a využívá pro ukládání dat databázový server MySQL. Systém běží na webovém serveru Apache HTTP Server.

Druhá verze je upravená aplikace Klub, kde je zanesen základní princip CRM. Ve druhé verzi se evidují zákazníci z řad klubů, spolků, asociací a také provozovatelé táborů. Reagovalo se tak na příchod novějších produktů Klub a Katalog dětských táborů.

[14] http://www.gwtproject.org/
[15] http://www.smartclient.com/product/smartgwt.jsp
[16] https://httpd.apache.org/

2.6 Portál České-školy.info

Tento portál je propojen s aplikací Smile a umožňuje školám nahrát hotové ŠVP z aplikace Smile na tento portál. Poté mohou školy vytvářet nad daným ŠVP studijní materiály pro výuku.

Obsahuje portlety, které byly vyvíjené pro portál Liferay, o kterém bude zmínka v následující kapitole.

3 Návrh optimalizovaných procesních postupů vývoje a používaných vývojových nástrojů

Proces vývoje ve firmě není formálně definován a vývoj probíhá podle aktuálních potřeb. Chyby, úkoly a požadavky (dále jen požadavky) jsou pořizovány z komunikace se zákazníkem nebo z interních zdrojů. Vše se zaznamenává do systému Redmine[17], což je open source řešení pro řízení projektů a bug tracking systém. Požadavky jsou přidělovány k dalšímu zpracování osobám pracujících na daných projektech. Požadavky jsou poté zpracovávány podle priority, verze aplikace a termínu dokončení. Zdrojové kódy jsou uchovávány v systému pro správu verzí Git[18]. Dříve se pro správu verzí používal systém CVS[19], který nyní slouží pouze k uchování historie vývoje před Gitem.

Zjistilo se, že tento stav je neudržitelný, protože se firma dostala do situace, kdy obchodní oddělení začalo generovat mnoho požadavků od zákazníků a požadovalo jejich vyřešení v co nejkratší době a přiřazovalo jim nejvyšší prioritu. Vývojáři na různých projektech byli zahlceni prací a nestíhali plnit přidělené požadavky. Proto se firma začala poohlížet po agilní metodice, kterou by šlo zavést do firemního prostředí.

3.1 Agilní metodiky

Ve světě informačních technologií existuje spousta postupů, jak vyvíjet software. Jedná se buď o rigorózní metodiky jako např. Vodopádový model, Spirálový model a RUP, anebo o agilní metodiky, které byly sjednoceny do Manifestu agilního vývoje softwaru (Manifesto for Agile Software Development) [1] světovými odborníky na softwarové inženýrství. Mezi známé agilní metodiky patří např. Extreme Programming (XP), Feature-Driven Development (FDD), SCRUM Development Process, Adaptive Software Development (ASD) nebo Test-Driven Development (TDD). Rozdíl mezi rigorózní a agilní metodikou určuje tzv. Váha metodiky [2].

Rigorózní metodiky bývají velmi robustní, protože velmi podrobně definují procesy, úkony a vyvíjený software. Při vývoji se tak klade velký důraz na podrobný popis softwaru, na plánování a řízení vývoje.

Cílem agilních metodik je fungující software, který přináší zákazníkovi užitek. Aby mohl být software rychle vyvinut a předán zákazníkovi k užívání, je proces vývoje omezen jen na základní principy a praktiky. Těmi jsou:

- **Orientace na zákazníka**, kde je potřeba se zákazníkem spolupracovat a dodat mu software, který chce a potřebuje.

[17] http://www.redmine.org/
[18] https://www.git-scm.com/
[19] http://www.cvshome.org/

- **Důraz na komunikaci**, kde je potřeba zajistit otevřenou a neformální komunikaci všech aktérů zahrnutých do vývoje software, a to na straně jak vývojového týmu, tak na straně zákazníka.
- **Jednoduchost a neformálnost**, kde je potřeba zjednodušit a zefektivnit proces vývoje, aby se snížily náklady a časová náročnost projektu.
- **Inkrementální vývoj s krátkými iteracemi**, kde je potřeba dodávat zákazníkovi software (nebo alespoň jeho prototyp) v krátkých časových úsecích.
- **Využívání moderních technologií**, kde se klade důraz na používání moderních programovacích jazyků a frameworků s využitím moderních vývojových prostředí.

3.1.1 Metodika FDD

Firma uvažuje o zavedení agilní metodiky FDD, která obsahuje principy, které se do určité míry již při vývoji používají.

Tato metodika využívá při vývoji software fáze návrhu a vývoje. Neustále se kontroluje stav projektu, dohlíží se na kvalitu v průběhu vývojového procesu a pravidelně se dodává nový software. S touto metodikou přišel Peter Coad, který spojil iterativní přístup s praktikami, které byly efektivní v průmyslu. Postupně se k němu přidali Jeff de Luca a Stephen Palmer, kteří Peteru Coadovi pomohli tuto metodiku zdokonalit a rozvést. První zmínka o FDD byla v publikaci Java Modeling in Color with UML [3]. Stephen Palmer ve své publikaci [4] mluví o tom, že FDD jde využít i při vývoji kritických aplikací.

Praktiky FDD

FDD je postaveno na základní skupině praktik. Tyto praktiky nejsou nové, ale její pozitiva spočívají ve správné kombinaci těchto praktik, které se mezi sebou doplňují a ovlivňují. Základní praktiky jsou [4]:

- **Doménové objektové modelování** (Domain Object Modeling) - jedná se o abstraktní objektový model, který vývojáře seznamuje s doménou a uvádí je do problematiky. Zachycuje se chování objektů doplněné o abstraktní sekvenční diagramy znázorňující, jak spolu objekty komunikují. Detailní architektura se rozebírá až v dalších fázích.
- **Vývoj podle užitných vlastností** (Developing by Feature) - jedná se o specifikaci funkčních požadavků, které jsou zapsané tak, aby jim zákazník rozuměl. Zákazník se tak může podílet na postupu projektu. Užitná vlastnost (feature) je malá funkčnost s hodnotou pro zákazníka, která se zapisuje ve formátu <akce> <předmět> <podrobnosti>. *Akce* označuje činnost, *předmět* je artefakt, na kterém je akce prováděna a *podrobnosti* upřesňují vlastnost. Pokud užitnou vlastnost nejde vyřešit během jedné iterace, je vlastnost

rozložená na menší vlastnosti. Tím zákazník vidí měřitelný postup na projektu. Užitná vlastnost je mapována k určité činnosti v rámci podnikového procesu.

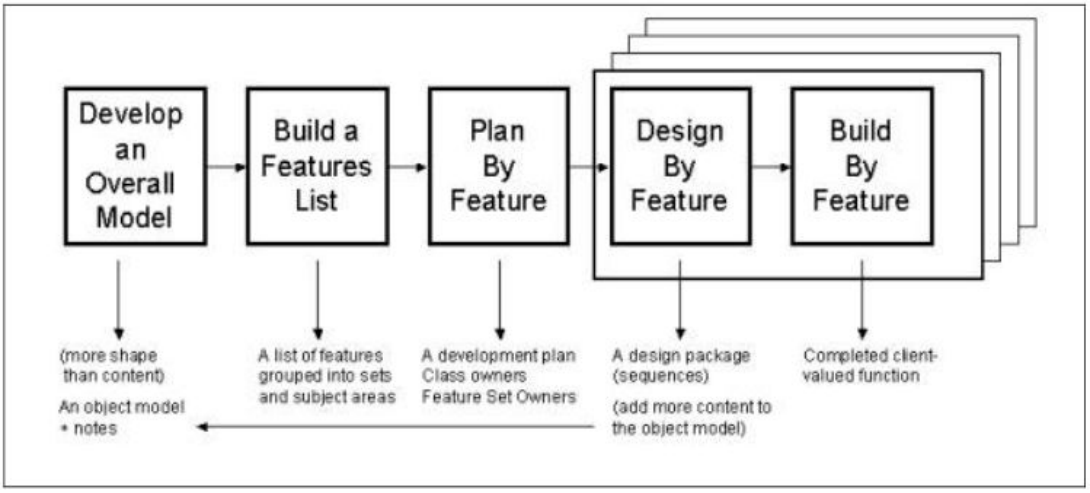
- **Vlastnictví tříd** (Individual Class Ownership) - jedná se o určení vlastníka třídy, který je zodpovědný za celou třídu. Nejedná se pouze o zajištění správného fungování samotné třídy, ale i o zajištění komunikace dalších tříd s touto třídou, aby byla zajištěna integrita systému. Vlastník jako autor zná třídu nejlépe a je tak schopen rychleji reagovat na změny.
- **Týmy pro užitné vlastnosti** (Feature Teams) - jedná se o vytvoření týmu pro užitnou vlastnost, kde jsou seskupeni vlastníci tříd. Ti jsou zodpovědní za vývoj dané užitné vlastnosti. Třídy vycházejí z doménového objektového modelu, kterým byli přiřazeni vlastníci tříd.
- **Inspekce** (Inspection) - jedná se o kontrolu a opravu chyb, které vznikají při návrhu i při vývoji.
- **Pravidelné buildy** (Regular Builds) - jedná se o pravidelné sestavování hotových užitných vlastností. Tato praktika umožňuje zákazníkovi otestovat užitnou vlastnost a vyjádřit se tak, zda byla vlastnost splněna. Může se i odhalit problém s nasazováním aplikace a tento problém před odevzdáním zákazníkovi vyřešit.
- **Řízení konfigurací** (Configuration Management) - jedná se o správu verzí produktu, kde se spravuje specifikace požadavků, modely, zdrojové kódy a další artefakty související s vydanou verzí produktu.
- **Reportování/Viditelnost výsledků** (Reporting/Visibility of Results) - jedná se o minimalizaci činností jako sbírání informací o stavu projektu, porady o stavu projektu a nepotřebné formuláře. Zaměřuje se na jednoduchý, efektivní a nejlépe automatický sběr těchto informací.

Procesy FDD

FDD sice popisuje vývoj softwaru ve formě procesu, ale nezasahuje nijak do hloubky. Jedná se o obecný rámec, jak má být zhruba software vyvíjen, ale jak toho vývojář dosáhne, nechává na vývojáři samotném. Pro popis procesu se používá vzor ETVX (Entry, Task, Verification, eXit), který u každého procesu specifikuje jasná vstupní kritéria, vyjmenuje úkoly, určí nástroje pro verifikaci a definuje výstupní podmínky procesu. Na obrázku 1 je znázorněn jednoduchý přehled procesů popsaných v metodice FDD. Poslední dva procesy probíhají zpravidla ve dvoutýdenních iteracích.

Mezi procesy patří [4]:

- **Vypracování celkového modelu** (Develop an Overall Model)



Obrázek 1: Procesy FDD

- *Vstupním kritériem* je výběr doménového experta, hlavního programátora a hlavního architekta.
- *Úkolem* je sestavení týmu pro vytvoření modelu, podrobně prozkoumat doménu, nastudovat potřebné dokumenty, vytvořit a odladit globální model a sepsat komentáře k modelu.
- *Verifikace* může probíhat formou interních (uvnitř týmu) nebo externích (zákazníkem) řízení.
- *Výstupním kritériem* je vytvořený objektový model - diagram tříd s identifikovanými atributy a metodami, sekvenční diagramy, detaily zachycené v komentářích.

- **Sestavení seznamu užitných vlastností** (Build a Features List)
  - *Vstupním kritériem* je vytvořený celkový model.
  - *Úkolem* je sestavení týmu pro vytvoření seznamu užitných vlastností, vytvoření užitných vlastností a zařazení užitných vlastností do množin užitných vlastností (feature sets).
  - *Verifikace* může probíhat formou interních (uvnitř týmu) nebo externích (zákazníkem) řízení.
  - *Výstupním kritériem* jsou množiny a seznamy užitných vlastností, definice minimální množiny užitých vlastností akceptovaný zákazníkem, seznam oblastí domény, pro každou oblast je vytvořen seznam business procesů týkající se domény, identifikace užitných vlastností pro každý krok business procesu.

- **Plánování užitné vlastnosti** (Plan By Feature)
  - *Vstupním kritériem* je sestavený seznam užitných vlastností.

- *Úkolem* je sestavit tým pro plánování, sestavit priority a pořadí, v jakém se užitné vlastnosti budou vyvíjet, přiřazení business procesů hlavním programátorům, přiřazení třídy vlastníkům tříd.
- *Verifikace* může probíhat formou průběžných interních řízení.
- *Výstupním kritériem* je plán vývoje - přibližná data dokončení pro všechny business procesy, hlavní programátoři jsou přiřazeni k business procesům, rozdělení seznamu užitných vlastností na oblasti podle business procesů, seznam tříd s jejich vlastníky.

- **Návrh užitné vlastnosti** (Design By Feature)
  - *Vstupním kritériem* je dokončený proces Plánování užitných vlastností.
  - *Úkolem* je sestavení týmu pro realizaci dané užitné vlastnosti z vlastníků tříd, může se zařadit zkoumání domény a doprovodných dokumentů, vytvoření sekvenčního diagramu pro danou užitnou vlastnost, odladit objektový model, sepsat hlavičky tříd a metod.
  - *Verifikace* probíhá formou inspekce návrhu.
  - *Výstupním kritériem* je vytvořený balíček návrhu (Design Package) - obecné informace o užitné vlastnosti, diagramy posloupností, případné další možnosti v návrhu, objektový model dané užitné vlastnosti, hlavičky tříd a metod, harmonogram prací pro vývoj užitné vlastnosti.

- **Realizace užitné vlastnosti** (Build By Feature)
  - *Vstupním kritériem* je dokončený návrh užitné vlastnosti (balíček návrhu).
  - *Úkolem* je vývoj třídy a metody pro danou užitou vlastnost, provést inspekci napsaného kódu, testování, příprava užitné vlastnosti na integraci a sestavení.
  - *Verifikace* probíhá formou průběžné inspekce kódu, testování.
  - *Výstupním kritériem* je úspěšná inspekce třídy a jejích metod, úspěšná komunikace mezi třídami, dokončený vývoj užitné vlastností.

3.2 Zavádění FDD ve firmě

Po nastudování FDD se ukázalo, že některé praktiky z FDD se ve firmě již používají. Využívá se doménové objektové modelování pro znázornění základní domény. Užitné vlastnosti jsou v případě firmy požadavky v systému Redmine, které jsou seskupovány do verzí a do souhrnných požadavků. Ve firmě působí pět vývojářů, kteří se podílejí na vývoji a údržbě aplikací. Na každém projektu se podílí několik vývojářů a jsou zodpovědní za část projektu. Každý vývojář je tak vlastníkem určité části, za kterou zodpovídá. Vývojáři mají v řešení požadavků volnost, důležité je splnění požadavků. Inspekci kódu a modelu provádí vývojáři také sami. Pro správu verzí se využívá Git.

Při vývoji se nevyužívá dvoutýdenní iterace, ale vyvíjí se tak dlouho, dokud požadavek není vyřešen. Požadavky nejsou dostatečně plánovány a neprobíhají automatické buildy. Tyto nedostatky může firma vyřešit právě zavedením FDD.

Vedení firmy požaduje využít stávající systém Redmine a nastavit jej tak, aby proces řešení požadavků vycházel z FDD a aby požadavky podléhaly plánování. Požadavky vznikají jak uvnitř firmy, tak od zákazníků přes helpdesk. Proto jsem v rámci své diplomové práce dostal za úkol zpracovat proces řešení požadavků, který upřesňuje postup řešení požadavků.

3.2.1 Proces řešení požadavků

Cílem je v systému Redmine nastavit proces tak, aby řešení požadavků bylo řízené a za požadavky byl zodpovědný projektový manažer, který na plnění požadavků bude dohlížet a korigovat příchozí požadavky.

Navržený proces je znázorněn na obrázku 24 (Příloha A). Oznamovatel zaznamená do systému Redmine příchozí požadavek z komunikace se zákazníkem nebo navržený zaměstnancem firmy. Do systému Redmine se zaznamenají pouze ty požadavky, které není možné obratem (přímou odpovědí) vyřešit. Manažer ověří, zda je požadavek oprávněný a je třeba ho řešit. Pokud se jedná o opravu, přiřadí manažer požadavek řešiteli, který je vlastníkem tříd v dané oblasti požadavku. V případě že se jedná o větší zásah do projektu nebo požadavek výrazně ovlivňuje funkcionalitu je potřeba zahájit změnové řízení (Obrázek 26, Příloha A). Řešitel buď požadavek akceptuje a řeší, nebo se doptává na doplňující informace k požadavku, které v případě potřeby manažer vykomunikuje se zákazníkem. Po vyřešení požadavku kontroluje manažer splnění požadavku. Pokud požadavek není dostatečně vyřešen, je znovu otevřen a přiřazen řešiteli k dopracování. V případě pozitivní kontroly je požadavek uzavřen. Pokud přišel požadavek od zákazníka, je zákazník před uzavřením obeznámen o stavu požadavku a harmonogramu nasazení. K tomuto procesu byl navržen a schválen stavový diagram znázorněný na obrázku 25 (Příloha A), kterým se budou řídit stavy požadavků v systému Redmine.

Samotné změnové řízení na obrázku 26 (Příloha A) probíhá tak, že se požadavky vyžadující změnového řízení vloží do jednoho rodičovského požadavku. Zašle se upozornění zainteresovaným zodpovědným osobám. V případě změnového řízení je potřeba informovat zejména osobu zodpovědnou za požadavky na aplikaci, analytika aplikace, designéra aplikace a vývojáře. Jednotlivé osoby se k požadavkům vyjádří a poté se rozhodne, zda se daný požadavek odloží nebo se na něm začne pracovat. V případě potvrzení prací manažer vytvoří doplňující úkoly, které jsou potřeba vyřešit v rámci změnového řízení. Jakmile jsou doplňující úkoly vyřešeny, provede se kontrola změnového řízení. Pokud je vše v pořádku, je změnové řízení uzavřeno. Úkoly ve změnovém řízení se řídí vlastním stavovým diagramem znázorněný na obrázku 27 (Příloha A).

3.3 Vývojové nástroje

Jako hlavní programovací jazyk zůstává ve firmě Java a vývojové prostředí Eclipse IDE. Z databázových serverů se bude primárně používat MySQL a aplikační server zůstane Apache Tomcat. Cílem je zpřístupnit všechny současné a budoucí aplikace přes webový prohlížeč. Nově bude vývoj určen pro webový portál Liferay[20], který hojně využívají velké společnosti na českém trhu jako Česká pojišťovna, Česká spořitelna nebo T-Mobile. Pro vývoj prezentační vrstvy se bude používat JSF a JSF komponenty Primefaces[21]. Správu knihoven a o build aplikací se postará Apache Maven[22]. O automatické buildy a o nasazení aplikací se postará nástroj Jenkins[23].

3.3.1 Liferay

Liferay je volně dostupný open source portál vydávaný jako Community Edition (CE). Liferay nabízí i placenou podporu[24], která zahrnuje například podporu pro řešení incidentů, záplaty, nové funkčnosti nebo další služby pro platící zákazníky vydávaný jako Enterprise Edition (EE). Záplaty pro EE jsou po čase zařazeny a vydány i pro CE. Je napsaný v Javě a jedná se o webovou platformu s funkcemi běžně potřebných pro vývoj internetových stránek a portálů. Umožňuje správu uživatelů, organizací, internetových stránek, dat, aplikací a procesů z jednoho centrálního uživatelského rozhraní. Instalace je dostupná pro aplikační server Apache Tomcat a díky různým konektorům jej lze připojit při instalaci k nejběžnějším databázovým serverům. Portál Liferay lze nakonfigurovat a upravit podle vlastních potřeb. Je možné do něj přidávat další aplikace pomocí portletů, vytvořit vlastní design nebo použít existující. Jeho velkou výhodou je i to, že lze na jednom serveru provozovat nespočet různých instancí s různou konfigurací a mnoho webových stránek na různých adresách.

Liferay portlet je rozšířený Java portlet a jedná se o webovou komponentu. Portlet umožňuje integraci do portálu, který je založený na Java EE a jeho API je definováno v Java Portlet Specification[25]. Portlet rozšiřuje uživatelské rozhraní portálu nebo jeho funkčnost.

Portál Liferay má k dispozici rozšíření Liferay IDE, které umožňuje vyvíjet a testovat portlety přímo v prostředí Eclipse. Součástí Liferay IDE je i Service Builder, což je nástroj na generování servisní vrstvy pro ORM, který odděluje objektový model od databáze. Objekty mohou být namapovány na existující tabulky v databázi nebo nemusí být objekt ukládán do databáze vůbec. Vše se upravuje v XML souborech a po vygenerování se vytvoří SQL skripty a funkce pro aplikační vrstvu. Nástroj v základu generuje CRUD dotazy nad objekty, ale lze jej rozšířit o další dotazy s konkrétními parametry. V aplikační vrstvě se přistupuje k databázi přes služby, které mohou být přístupné lokálně uvnitř portálu, nebo z venku pomocí webových služeb. Vytváření

[20] http://www.liferay.com/
[21] http://www.primefaces.org/
[22] https://maven.apache.org/
[23] https://www.jenkins.io/
[24] https://www.liferay.com/downloads/liferay-portal/overview
[25] http://www.oracle.com/technetwork/java/jsr286-141866.html

tabulek a indexů v databázi řeší portál Liferay sám po nasazení portletu a perzistenci databáze zajišťuje portál Liferay taktéž sám.

3.3.2 JSF a Primefaces

JSF je technologie, která zavádí standardy[26] pro tvorbu uživatelského rozhraní na straně serveru, čímž se zjednodušuje vývoj webových aplikací. Je navržený tak, aby zajistil vývoj uživatelského rozhraní nezávisle na značkovacím jazyce, protokolu a klientském zařízení. Odděluje tak uživatelské rozhraní od aplikační vrstvy. Aplikační vrstva je tvořena pomocí tříd Java Beans.

Primefaces je open source soubor komponent, který rozšiřuje framework JSF, jenž umožňuje rychlý vývoj sofistikovaných webových aplikací. Obsahuje mnoho komponent pro grafické uživatelské rozhraní, včetně komponent upravených pro mobilní zařízení. Umožňuje datovým komponentám provádět třídění, filtrování a stránkování. Komponentám lze nadefinovat vlastní vzhled. Zlepšuje a zjednodušuje propojení technologie AJAX s JSF.

3.3.3 Apache Maven

Liferay IDE umožňuje spravovat knihovny a závislosti pomocí nástroje Apache Maven. Pomocí toho nástroje se provádí buildování aplikace. Určuje, jak má být aplikace kompilována a dále popisuje závislosti na knihovnách. Ty jsou definovány pomocí POM v konfiguračním souboru `pom.xml`. Knihovny poté stahuje z veřejně dostupných repositářů do lokálního repositáře bez nutnosti hledání knihovny na stránkách autora.

3.3.4 Jenkins

Jenkins je open source kontinuální integrační nástroj napsaný v Javě, jenž zrychluje vývoj aplikací pomocí automatizovaných postupů. Řídí a kontroluje procesy vývoje, jako jsou pravidelné buildy, testování, nasazení aplikací a další. Jenkins může být nastaven tak, aby kontroloval změny zdrojových kódů v Gitu, provedl automatický build pomocí Apache Maven, provedl testování a pokud testování proběhlo bez chyb, provedl nasazení na produkční prostředí.

[26] http://www.oracle.com/technetwork/java/javaee/overview-140548.html

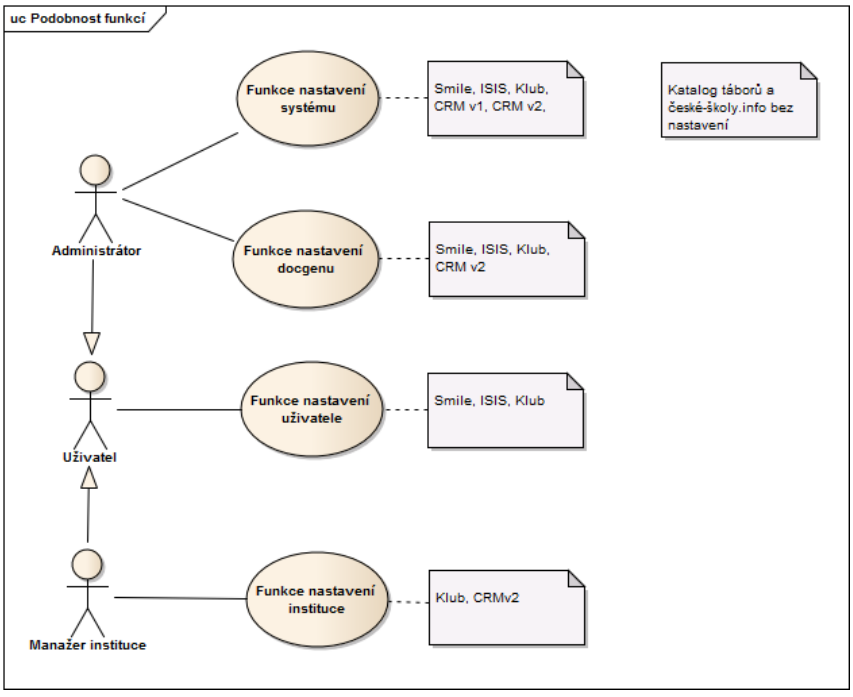
4 Analýza stávajících produktů a identifikace integrovatelných částí

Součástí analýzy bylo nastudování technické a uživatelské dokumentace [6, 7, 8, 9, 10, 11, 12, 13, 14] současných aplikací, abych se seznámil s jejich strukturou a funkcionalitou. A to z důvodu identifikace podobných částí, které jsou pro aplikace společné. Jedná se o základní části, které se standardně používají ve všech současných aplikacích a mohou se objevit i v aplikacích nových.

Firma dále vznesla požadavek, že i když se budou všechny aplikace slučovat do jednoho prostředí, z důvodu náročnosti jak časové, tak i finanční budou současné aplikace ponechány v současném stavu a budou se převádět v jiné etapě. Nové moduly rozšiřující tyto aplikace budou vyvíjený v novém prostředí a budou s původními aplikacemi provázány.

4.1 Nastavení

Po rozsáhlém zkoumání všech aplikací byla nalezena podobnost funkcí, které řeší různá nastavení chování aplikací pro určitou oblast. Pro každé takové nastavení existuje datová struktura s vazbou na konkrétní tabulku, která je na nastavení závislá. Tento stav vznikl díky kopírování základní struktury do dalších aplikací. Poté se musely funkce přizpůsobit danému frameworku. Podobné funkce jsou znázorněné na obrázku 2.



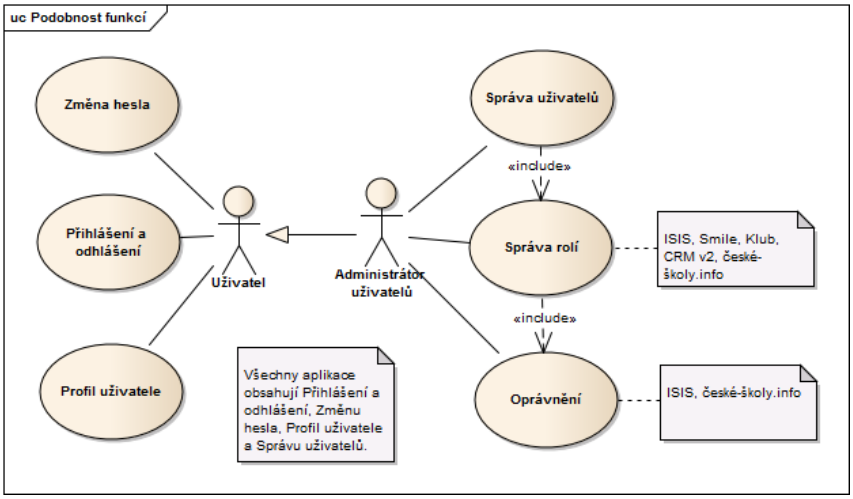
Obrázek 2: Funkce nastavení v aplikacích

Po úvaze, že další nastavení mohou přibývat a pro každé nové nastavení je zbytečné vytvářet nové funkce a datové struktury, jsem se rozhodl tento stav zjednodušit a společnou funkcionalitu převést do jednoho portletu Settings (Kapitola 5.2.1).

4.2 Uživatelé

Přihlašování do aplikací probíhá obvyklým způsobem po zadání unikátního přihlašovacího jména a hesla. V aplikaci Katalog táborů nebo v první verzi CRM je správa uživatelů jednoduchá. V ostatních aplikacích se už využívá princip rolí uživatelů a práv logických a systémových objektů, které omezují chování uživatelů v aplikacích. U uživatelů jsou vedeny kontakty, které jsou popsané samostatně kapitole 4.4.

Vzhledem k tomu, že aplikace jsou vyvíjeny v různých frameworcích, základní struktura a funkcionalita zůstala stejná. Rozdíl nastal v tom, že se pro každou aplikaci muselo naprogramovat samostatné rozhraní pro každý framework. Každá aplikace potom využívá správu uživatelů rozdílně. Například v aplikaci Smile jsou role a práva definována napevno v kódu bez možnosti editace práv uživatelem, v aplikaci ISIS může role a práva, která se ukládají do databáze, upravovat uživatel s potřebnými právy. Podobné funkce napříč aplikacemi jsou znázorněné na obrázku 3.



Obrázek 3: Funkce uživatelů

Portál Liferay má správu uživatelů dobře propracovanou, obsahuje role i práva, které se dají definovat pro portál, pro webové stránky i pro portlety. Jelikož správa uživatelů nesplňuje veškeré požadavky firmy, je možné díky dostupným funkcím z portálu správu uživatelů přizpůsobit vlastním potřebám ve vlastním portletu. O tom pojednává kapitola 5.2.1.

4.3 Instituce

V každé aplikaci jsou vedeny informace o zákazníkovi, které jsou potřebné např. pro výstupy generované aplikacemi. Tyto údaje jsou evidovány ve struktuře jako instituce. V aplikaci Katalog táborů reprezentuje instituci pojem organizace, ale jedná se o stejný princip. U instituce jsou vedeny kontakty, které jsou popsané samostatně v kapitole 4.4.

Podobně jako u uživatelů je struktura a funkcionalita instituce stejná a pro každý framework je definováno jiné rozhraní. Například v aplikaci Smile se jedná o identifikační údaje navázané na ŠVP, zákazník aplikace Klub může mít přístup minimálně k jedné instituci, kterou může spravovat.

Portál Liferay pracuje se strukturou nazývající se organizace, která je totožná s institucí ve firemních aplikacích. Portál Liferay je vyvíjen tak, aby se v něm dala vytvářet organizační struktura velkých korporací. Obsahuje správu organizací, kde lze přiřazovat uživatele s příslušnými právy k jednotlivým organizacím. Avšak portál Liferay není přizpůsoben pro Českou republiku. Chybí například IČ, DIČ, což jsou údaje potřebné pro fakturaci. Proto je i tato část rozšířená stejně jako správa uživatelů v Portletu OU (Kapitola 5.2.1) o další funkcionalitu nad organizacemi.

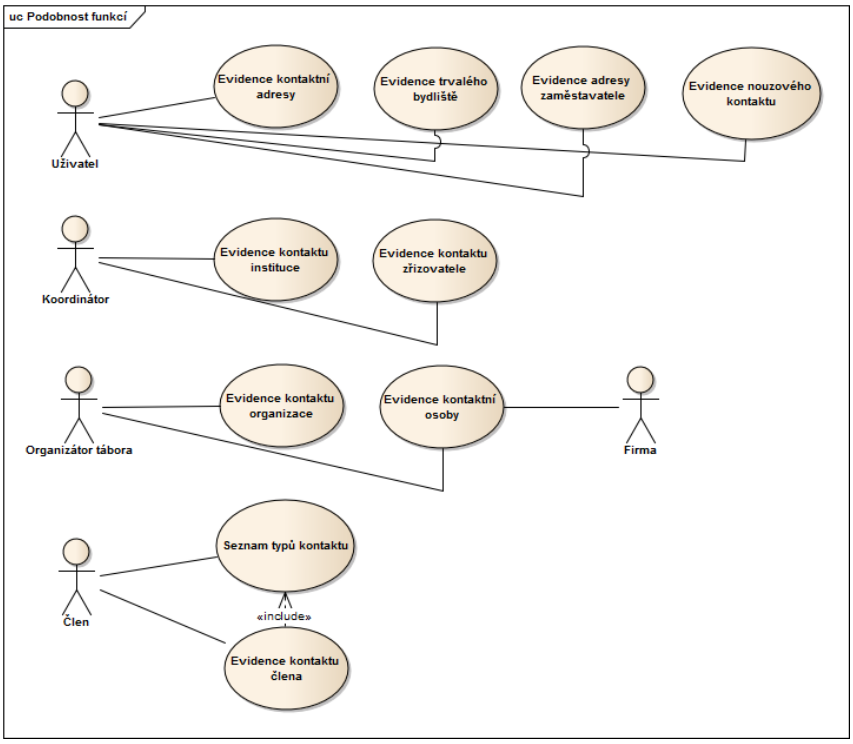
4.4 Kontakty

Kontakty se evidují ve všech aplikacích u uživatelů a institucí. Také se kontakty evidují u osob a firem v obou verzích CRM, avšak struktura je rozdílná oproti kontaktům uživatele a instituce. V aplikaci Katalog táborů jsou kontakty evidovány přímo v tabulkách tábora a organizace. Některé kontakty jsou vázány přímo nebo mají přidělený typ kontaktu. V některých případech není adresa součástí kontaktu. V některých případech obsahuje uživatel či instituce atributy, které odkazují na konkrétní kontakt, např. zřizovatel či kontakt na zaměstnavatele, což komplikuje identifikaci kontaktu v aplikaci. Významným kontaktním údajem v dnešní době jsou sociální sítě, pro které není žádná struktura definována v žádné aplikaci. Druhy kontaktů pořizované ve všech aplikacích jsou znázorněné na obrázku 4.

Řešení kontaktů v portále Liferay nám kompletně nevyhovuje, proto jsem se rozhodl nahradit tyto kontakty vlastním řešením, které eliminuje neduhy ze současných aplikací. Struktura a funkčnost bude řešena v rámci Portletu OU (Kapitola 5.2.1), kde bude definováno rozhraní s dostupnými funkcemi.

4.5 Číselníky

V jednotlivých systémech se nachází mnoho datových struktur, které slouží k ukládání hodnot číselníků. Aby se mohl číselník přidat do aplikace, musí se celý samostatně naprogramovat a nakonfigurovat. Některé hodnoty jsou uložené ve výčtových typech a nejde je bez kompilace aplikace upravovat. Struktury se buď liší pouze v pojmenování, nebo obsahují některé atributy



Obrázek 4: Funkce kontaktů

navíc. Jedná se například o typ školy, rodinný stav, číselníky územních celků, typ člena, typ tábora a další.

Stejně jako u různých nastavení se princip číselníků zanese do samostatného Portletu LOV (Kapitola 5.2.1), který bude spravovat číselníky pro všechny portlety a bude dostupný přes jedno rozhraní.

4.6 Generování dokumentů

O generování dokumentů v aplikacích se stará modul Docgen. Pro každý framework musel mít definované vlastní rozhraní. V tomto modulu se vystřídalo od roku 2005 mnoho technologií, které postupem času zastaraly. Ze začátku se používaly knihovny Apache Jelly[27] na generování XML souborů, Apache Velocity[28] na generování textových souborů a HTML a Apache FOP[29] využívající XSL pro generování dokumentů do PDF. Apache Velocity byl později nahrazen knihovnou Apache FreeMarker[30]. Apache FOP byl pro změnu nahrazen knihovnou JasperReports[31].

Bylo rozhodnuto, že pro portál Liferay bude tento modul zjednodušen. Zůstane zachován Apache FreeMarker, jenž bude použit pro generování jednoduchých výstupů v HTML a XML.

[27] http://commons.apache.org/proper/commons-jelly/
[28] http://velocity.apache.org/
[29] http://xmlgraphics.apache.org/fop/
[30] http://www.freemarker.org/
[31] http://community.jaspersoft.com/project/jasperreports-library/

Pro složitější typy výstupů bude použit DocBook[32], díky kterému bude možné vygenerovat dokumenty do HTML, PDF a dalších formátů. Struktura a funkčnost bude řešena v rámci Portletu DocGen (Kapitola 5.2.1).

4.7 Události

S událostmi se pracuje pouze v aplikaci Katalog táborů. Událost zde reprezentuje tábor a jeho turnusy. Jelikož je Katalog táborů naprogramovaný v PHP, je potřeba modul události přepsat do Javy. U této příležitosti vzešel požadavek na vytvoření obecné události, kde budou tábor a turnusy rozšířením obecné události. Uvažuje se o vývoji dalších funkcí rozšiřující událost např. pro sportovní akce, školní akce nebo workshopy.

Tábory a turnusy se budou převádět do nového systému. Struktura a funkčnost událostí bude řešena v rámci Portletu Events (Kapitola 5.2.2).

[32] http://www.docbook.org/

5 Návrh procesu integrace vývoje produktů

Základním portálem, do kterého se budou produkty integrovat, bude portál Pohodlné.info. V první řadě dojde k vývoji a nasazení základních portletů (Kapitola 5.2.1), které jsou společné pro všechny existující a budoucí produkty. Dále dojde k vývoji a nasazení hlavních portletů (Kapitola 5.2.2), které jsou závislé na základních portletech. V první řadě dojde na integraci portálu Pohodlné.info s aplikací Klub. Celý převod aplikace Klub je plánován do jiné etapy.

Současně dojde k přesunu běžícího portálu České-školy.info pod portál Pohodlné.info. Existující portlety běžící na portále České-školy.info budou upraveny tak, aby byly navázány na nové základní portlety. Etapy převodu aplikací do jednoho portálu jsou popsány v kapitole 7.

5.1 Integrace se stávajícími produkty

Jelikož nejde všechny aplikace převést najednou do nového portálu, bude potřeba provést integraci se stávajícími produkty, aby se mohly vyměňovat data mezi aplikacemi. Je proto důležité zvolit vhodné řešení, které bude využito pro integraci portálu Pohodlné.info s existujícími aplikacemi. Způsobů, jak se aplikace budou mezi sebou integrovat je celá řada, jak je uvedeno v publikaci Enterprise integration patterns [5].

5.1.1 Způsoby integrace

Existují různé způsoby, jak integrovat aplikace, které se vyvíjely v průběhu času. Podle [5] lze tyto přístupy shrnout do čtyř hlavních způsobů integrace:

- **Přenos souborů** (File Transfer) - Aplikace produkuje soubory se sdílenými daty a jiné aplikace tyto soubory konzumují.
- **Sdílená databáze** (Shared Database) - Aplikace ukládají data do společné databáze.
- **Vyvolání vzdálené procedury** (Remote Procedure Invocation) - Aplikace mají vystavené některé své procedury, ke kterým lze přistupovat vzdáleně. Jiné aplikace se s těmito procedurami spojí a provádí se výměna dat skrze tyto procedury.
- **Zasílání zpráv** (Messaging) - Aplikace se připojuje ke společnému systému zasílání zpráv a výměna dat probíhá pomocí zpráv.

Přenos souborů a sdílená databáze umožňuje aplikacím sdílet svá data, ale nikoliv jejich funkčnost. Vyvolání vzdálené procedury umožňuje sdílení funkčnosti, ale musí být navázáno pevné spojení. Častým úkolem integrace je vytvoření spolupráce mezi různými systémy, nikoliv je spojit dohromady.

Přenos souborů zajišťuje, že aplikace budou oddělené, ale na úkor času. Systémy nemohou mezi sebou držet krok, protože spolupráce mezi nimi je příliš pomalá. Sdílená databáze uchovává

společná data, ale za cenu neustálého připojení k jedné databázi s rizikem nekonzistence dat. I když se zdá vyvolání vzdálené procedury rozumným řešením, přináší spoustu nedostatků. Jedním z nich je skutečnost, že změnou modelu aplikace dojde k roztříštění aplikace. Dalším nedostatkem je pomalé volání procedur i přes to, že se jeví jako lokální. Největším nedostatkem je ale ta skutečnost, že pokud dojde k chybě v jedné aplikaci, bude to mí vliv na chod ostatních aplikací.

Zasílání zpráv reaguje na problémy spojené s předešlými třemi způsoby. Zasílání zpráv nevyžaduje, aby systémy běžely a byly dostupné v jednu dobu. Aplikace mohou mít zcela odlišné koncepční modely. Častým zasíláním malých zpráv umožňuje aplikacím spolupracovat na principu sdílení dat. Zasílání zpráv má výhodu v tom, že systémy jsou od sebe oddělené. Na druhou stranu tato nezávislost má za následek to, že integrační funkce musí být řešeny na obou stranách, aby vše správně fungovalo.

5.1.2 Integrace aplikací s Pohodlné.info

Některé aplikace běží společně na jednom serveru, ale jsou i aplikace, které běží i na různých serverech jak uvnitř sítě firmy, tak i na síti mimo firmu. Přenos souborů v tomto případě není použitelný kvůli rychlosti komunikace, jak bylo uvedeno výše. Sdílenou databázi nelze použít, protože aplikace mají oddělené databáze. Vyvolání vzdálené procedury také nelze použít z důvodu, že by aplikace musely mezi sebou udržovat perzistentní spojení, které by nám znemožnilo udržování aplikací nezávisle na sobě. Proto jedinou možností je využití webových služeb, jenž využívají princip zasílání zpráv.

Než se pustíme do integrace pomocí zasílání zpráv, je potřeba odpovědět na několik otázek podle [5]:

- **Jak se budou přenášet data?**

Odesílatel odesílá data příjemci zasláním zprávy přes Kanál zpráv (Message Channel), který spojuje odesílatele s příjemcem.

Existují dva hlavní způsoby, jak kanál zpráv využít. Jedním způsobem je Point-to-point channel, který odesílá data k jinému příjemci ve spojeni 1:1. Druhým způsobem je Publish-Subscribe channel, který odesílá data tak, že je nakopíruje pro každého příjemce ve spojení 1:N. Pro firemní účely, kdy se bude portál Pohodlné.info dotazovat na konkrétní data v jiné aplikaci, bude zasílání zpráv probíhat způsobem 1:1, čili přes Point-to-point channel.

Zprávy můžeme dělit podle záměru na Zprávy s příkazem (Command Message), která říká příjemci, jaká funkce se má u něj spustit. Dále je možné odeslat Zprávu s dokumentem (Document Message), což je zpráva obsahující data, které se mají na straně příjemce dále zpracovat. Dalším typem zprávy je Událost (Event Message), která posílá příjemci notifikace o změnách na straně odesílatele.

Když aplikace odešle zprávu, často očekává odpověď potvrzující zpracování zprávy a zpracovaný výsledek. Jedná se o Scénář typu požadavek-odpověď (Request-Reply scenario). Poža-

davkem je obvykle Zpráva s příkazem a odpovědí je Zpráva s dokumentem obsahující výslednou hodnotu nebo výjimku. Požadavek může obsahovat Návratovou adresu (Return Address), která příjemci řekne, kam se má odpověď vrátit. Může být také posláno několik požadavků v rámci jednoho procesu, které jsou rozšířené o Korelační identifikátor (Correlation Identifier), který určuje, jaká odpověď bude svázána s poslaným požadavkem.

Existují dva základní Scénáře typu požadavek-odpověď, které zahrnují Zprávu s příkazem v požadavku a Zprávu s dokumentem v odpovědi. Prvním scénářem je Zasílání zpráv přes RPC (Messaging RPC), kde požadavek nejenom vyvolá funkci na straně příjemce, ale chce vrátit návratovou hodnotu z funkce. Dalším scénářem je Zasílání zpráv přes dotaz (Messaging Query), kde je přes požadavek poslán dotaz příjemci, příjemce dotaz zpracuje a výsledky vrátí v odpovědi. To je způsob, jak aplikace využívají Zasílání zpráv k provedení dotazu vzdáleně.

Většinu těchto principů využívají webové služby, které využívají Scénáře požadavek-odpověď pro komunikaci mezi aplikacemi a bude potřeba přizpůsobit firemní aplikace tak, aby pracovaly s webovými službami.

- **Jak víme, kde se data budou posílat?**

Pokud odesílatel neví, kam posílat data, může data odeslat do Směrovače zpráv (Message Router), který data roztřídí podle kritérií a rozešle je ke správnému příjemci.

Jak už bylo popsáno výše, integrace portálu s aplikacemi bude řešená formou Point-to-point channel, proto nebude potřeba data třídit, protože budou zasílána k jednomu příjemci.

- **Jak víme, jaký formát dat bude použit?**

V případě, že odesílatel a příjemce nepoužívá totožný formát dat, Odesílatel zašle data do Překladače zpráv (Message Translator), který převede data do formátu, který přijímá příjemce.

V případě firmy půjde o data ve formátu JSON, proto nebude potřeba data převádět skrze překladač zpráv.

- **Jak připojit aplikaci do systému zasílání zpráv?**

Aplikace, která bude používat systém zasílání zpráv, musí mít implementované Koncové body zprávy (Message Endpoints), které budou zpracovávat odesílání a přijímání zpráv.

Portál Liferay v rámci Service Builderu umožňuje vytvářet webové služby, které lze využít k integraci externích aplikací s portálem Liferay. Aby se portál Liferay mohl propojit s ostatními aplikacemi, bude potřeba implementovat rozhraní pro webové služby do oněch aplikací.

5.2 Proces integrace vývoje produktů

Návrh procesu integrace vývoje znázorňuje obrázek 28 (Příloha B). Po provedení analýzy produktů a identifikaci integrovatelných části následuje implementace základních portletů popsaných v kapitole 5.2.1. Portlety závislé na základních portletech budou zařazené do hlavních portletů popsaných v kapitole 5.2.2.

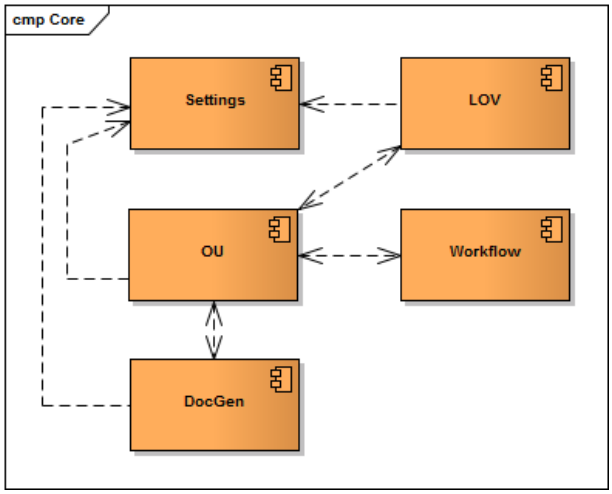
Testování portletů a realizaci integrace bude mít na starosti jiný zaměstnanec firmy a nebude součástí této práce.

Instalací testovacího a produkčního prostředí se myslí instalace portálu Liferay na testovací a produkční servery. Konfigurace se taktéž týká portálu Liferay na jednotlivých serverech. Tuto úlohu má na starosti jiný zaměstnanec firmy a nebude součástí této práce.

Návrh uživatelského rozhraní realizuje externí grafik. Implementace uživatelského rozhraní bude realizována jako samostatný nezávislý portlet typu theme pro portál Liferay. Vývoj bude mít na starosti jiný zaměstnanec firmy a nebude součástí této práce.

5.2.1 Základní portlety

Množina portletů, které budou základem systému, se bude nazývat Core. Tyto portlety budou na sobě závislé a budou obsahovat společnou funkcionalitu důležitou pro další rozšiřování systému. V případě dalšího rozšiřování systému se již nebudou tyto základní prvky znova programovat, pouze se využije to, co už bylo jednou naprogramované. Tím pádem to v budoucnu ušetří čas při dalším vývoji. Dále bude možné tyto portlety nasadit na další portály v případech, kdy bude poptávka od zákazníků na vytvoření vlastního portálu. Core se skládá z portletů Settings, List Of Values (LOV), Organizations and Users (OU), DocGen a Workflow, a je znázorněný na obrázku 5.



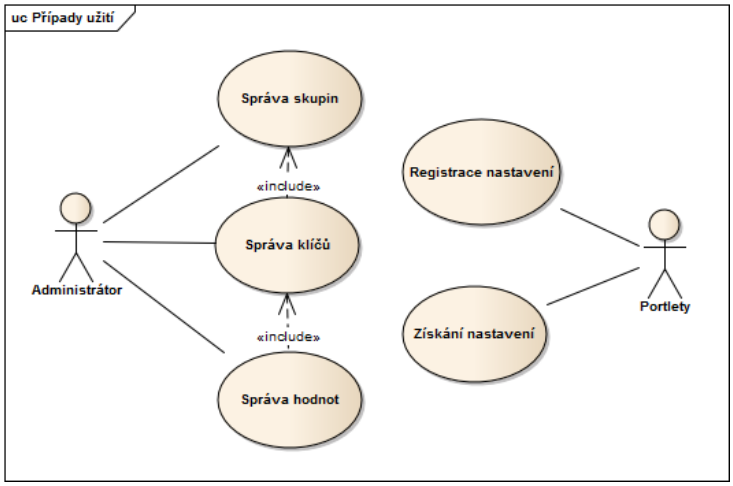
Obrázek 5: Struktura základních portletů

Základní portlety budou využity k sestavení stránek zařazené do správy portálu, které budou využívat uživatelé v příslušných rolích, převážně uživatel v roli Administrátor.

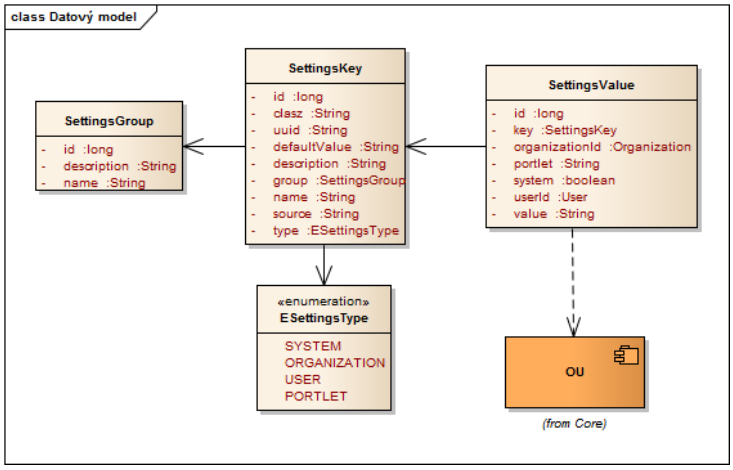
Portlet Settings

Portlet bude zajišťovat správu různých nastavení pro jednotlivé portlety, organizace či uživatele. Součástí portletu Settings bude i registrace definic nastavení do databáze. Bude se jednat

buď o systémové nastavení nebo nastavení patřící ostatním portletům, organizacím či uživatelům. Definice se budou registrovat z ostatních portletů při startu celého systému a bude tedy zajištěna jejich dostupnost a aktuálnost. Hodnoty nastavení se budou následně podle potřeb upravovat v průběhu provozu systému. Tento portlet bude dostupný pouze administrátorům portálu. Základní funkcionalita je znázorněná na obrázku 6.



Obrázek 6: Návrh základní funkcionality portletu Settings



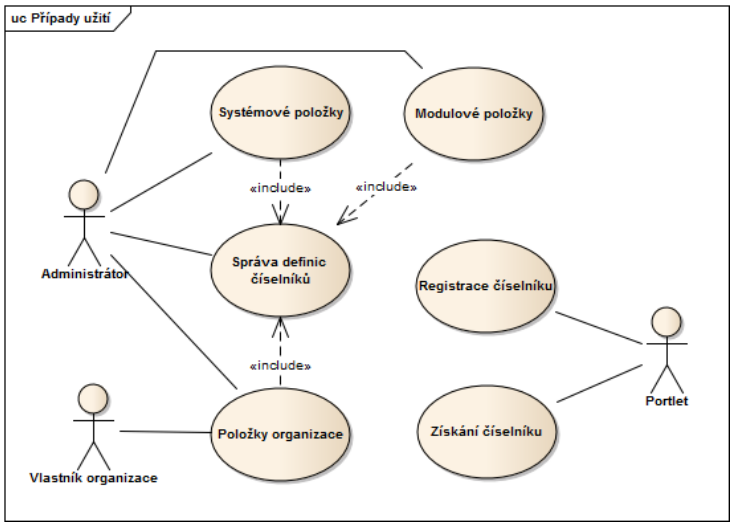
Obrázek 7: Datový model portletu Settings

Nastavení budou zařazená do skupin pro snadnou orientaci a vyhledávání. Eliminuje se tak implementace dalšího nového nastavení. Každé nastavení bude mít svůj unikátní klíč (UUID), podle kterého se bude v databázi vyhledávat konkrétní nastavení. Klíč bude rozdělený na typy podle významu použití klíče. Bude se jednat o systémový klíč, který bude ovlivňovat chování celého systému. Portletovým klíčem se bude ovlivňovat chování konkrétního portletu. Organizačním klíčem se bude ovlivňovat chování aplikace pro konkrétní organizaci a uživatelský klíč bude ovlivňovat konkrétního uživatele. Klíč bude obsahovat definici datového typu hodnoty, zdrojový

portlet, který klíč zaregistroval a popis. Dále klíč bude obsahovat výchozí nastavení, které se využije v případě, že neexistuje nastavení pro konkrétní portlet, organizaci nebo uživatele. Datový model portletu Settings je znázorněn na obrázku 7.

Portlet LOV

V tomto portletu se budou spravovat veškeré číselníky, které budou využívat jednotlivé portlety pro svou potřebu. Stejně jako portlet Settings bude mít i tento portlet registraci nových číselníků, která bude zajišťovat nahrání základních číselníků do databáze. Tento portlet bude dostupný pro uživatele v roli Administrátor a Vlastník organizace. Základní funkcionalita je znázorněná na obrázku 8.



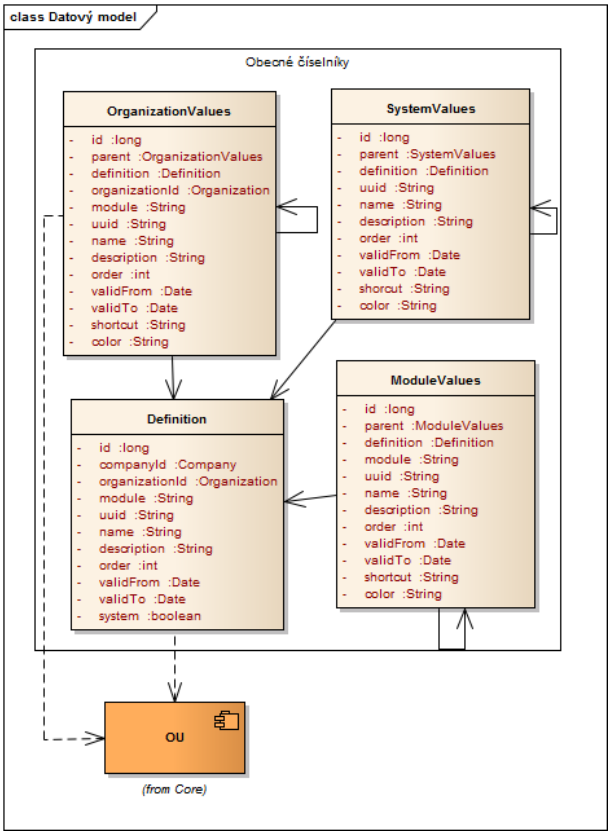
Obrázek 8: Návrh základní funkcionality portletu LOV

Číselníky se budou definovat ve správě definic číselníků, které spravuje Administrátor. Ten bude moci definice číselníků přidávat nebo je zneplatnit. Každá definice bude mít svůj unikátní identifikátor (UUID), podle kterého se bude definice vyhledávat. Definice číselníků budou rozděleny na systémové a nesystémové definice. Systémové definice jsou určeny výhradně pro číselníky, které jsou potřebné pro práci se systémem, a přístup k nim bude mít pouze Administrátor. S nesystémovou definicí bude moci pracovat i Vlastník organizace.

Každá definice číselníku bude obsahovat položky, které budou rozděleny na tři typy do samostatných tabulek. Položky budou uloženy ve stromové struktuře, kde položka může obsahovat i potomky. V první tabulce budou systémové položky, které budou dostupny všude tam, kde se definice číselníku bude používat. Příkladem může být měna, pohlaví nebo typy kontaktu. Systémové položky bude spravovat Administrátor a Vlastník organizace bude mít pouze právo čtení. V druhé tabulce budou modulární položky, kde budou uložené položky vyhrazené pro konkrétní modul, který bude rozšiřovat specifickou funkčnost. Jako příklad můžeme uvést typ kontaktu "Místo konání události", který se bude používat pouze v událostech. I tyto položky bude spra-

vovat Administrátor a Vlastník organizace bude mít pouze právo čtení. Ve třetí tabulce budou uloženy položky organizace, které bude spravovat i Vlastník organizace. Ten bude moci přidávat nové položky do definice číselníku a budou dostupné pro organizaci, která je bude vlastnit.

V jiných organizacích se tato položka nebude objevovat. Vlastník organizace bude mít možnost přejmenovat systémovou nebo modulární položku tak, že se nakopíruje položka se stejným klíčem mezi položky organizace, kde jí půjde přejmenovat. Pokud Vlastník organizace nebude chtít, aby se mu nějaké systémové nebo modulární položky nabízely, dojde ke zkopírování položky mezi položky organizace s ukončenou platností. U organizace se původní položka nebude zobrazovat, ale zobrazí se ostatním organizacím, které ji nechaly platnou. Položky budou mít vlastní UUID, jehož unikátnost se bude kontrolovat v rámci definice nad všemi třemi tabulkami v závislosti na platnosti a na vlastnictví organizace. Datový model portletu LOV je znázorněn na obrázku 9.



Obrázek 9: Datový model portletu LOV

Portlet LOV bude mít implementované rozhraní, které umožní jiným portletům získávat číselníky a položky z číselníků. První rozhraní vrací seznam položek číselníku podle UUID definice číselníku. Načtou se napřed systémové položky, poté modulární položky, a pokud v položkách organizace došlo k přejmenování či zneplatnění, dojde k načtení položky podle kritérií. Doplní se i ty položky, které byly přidané Vlastníkem organizace. Druhé rozhraní vrací konkrétní po-

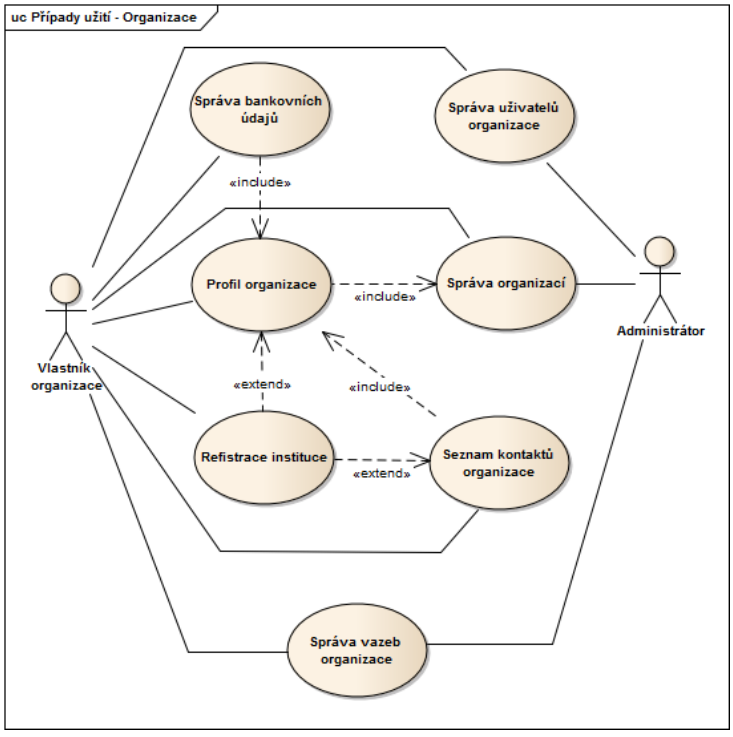
ložku číselníku. Podle UUID definice číselníku a UUID položky dojde k prohledání systémových, modulárních a organizačních položek a vrátí se aktuální položka v závislosti na platnosti a na konkrétní organizaci.

Portlet OU

Tento portlet bude rozšiřovat funkcionalitu uživatelů a organizací implementovanou v portále Liferay. Dále bude obsahovat funkcionalitu pro práci s kontakty. Proto bude portlet OU rozdělen na části Organizace, Uživatelé a Kontakty.

Organizace

Organizace bude mít rozšířenou strukturu o údaje, které nejsou dostupné v portále Liferay jako jsou IČ, DIČ a jiné. Organizace bude zařazená do kategorií, ve kterých působí (sport, školství, tábory atd.) Bude obsahovat typ (právní formu) organizace, aby bylo jasné, o jakou organizaci se jedná (z.s., s.r.o., o.p.s. atd.). Kategorie a typy organizace budou uloženy jako číselníky v portletu LOV (číselníky). Organizace bude moci spravovat své bankovní údaje, které budou potřeba při generování ekonomických výstupů. Základní funkcionalita uživatelů je znázorněná na obrázku 10.

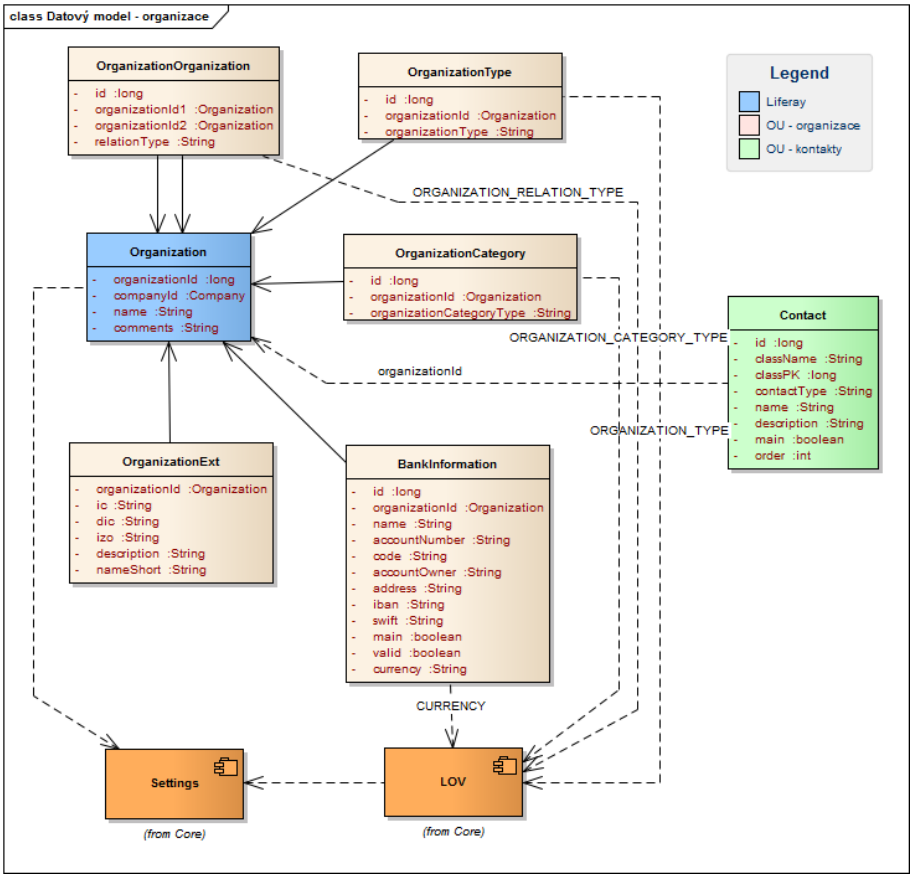


Obrázek 10: Návrh základní funkcionality organizací v portletu OU

V současné době jsou v portále Liferay organizace organizované do stromové struktury. Tento stav není dostačující, protože např. spolky mohou spadat pod více nadřazených organizací, anebo

se může jednat o spolupracující partnery. Proto bude potřeba pro organizace implementovat grafovou strukturu jejich vazeb. Vazba bude rozšířená o typ vazby z číselníků, která bude specifikovat formu propojení organizací. Vlastník organizace bude moci spravovat podřízené organizace.

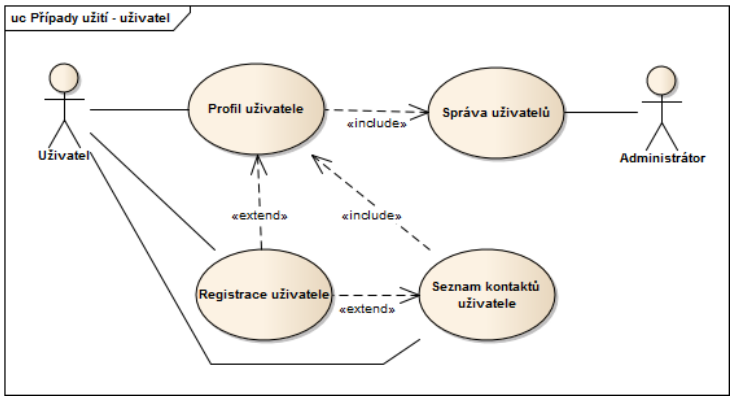
Zákazníci zaregistrují organizaci na portál pomocí registračního formuláře určený pro zákazníky. Portlet bude obsahovat správu organizací. Všechny organizace v systému bude spravovat uživatel v roli Administrátor. Vlastník organizace bude spravovat vlastní zaregistrované organizace. Vlastník organizace bude moci přiřazovat k organizaci uživatele, kteří pod organizaci pracují, např. trenéři, asistenti atd., aby mohli v rámci organizace provádět související úkony. Součástí portletu bude i katalog organizací členěný podle vyplněných kategorií. Datový model organizací v portletu OU je znázorněn na obrázku 11.



Obrázek 11: Datový model organizací v portletu OU

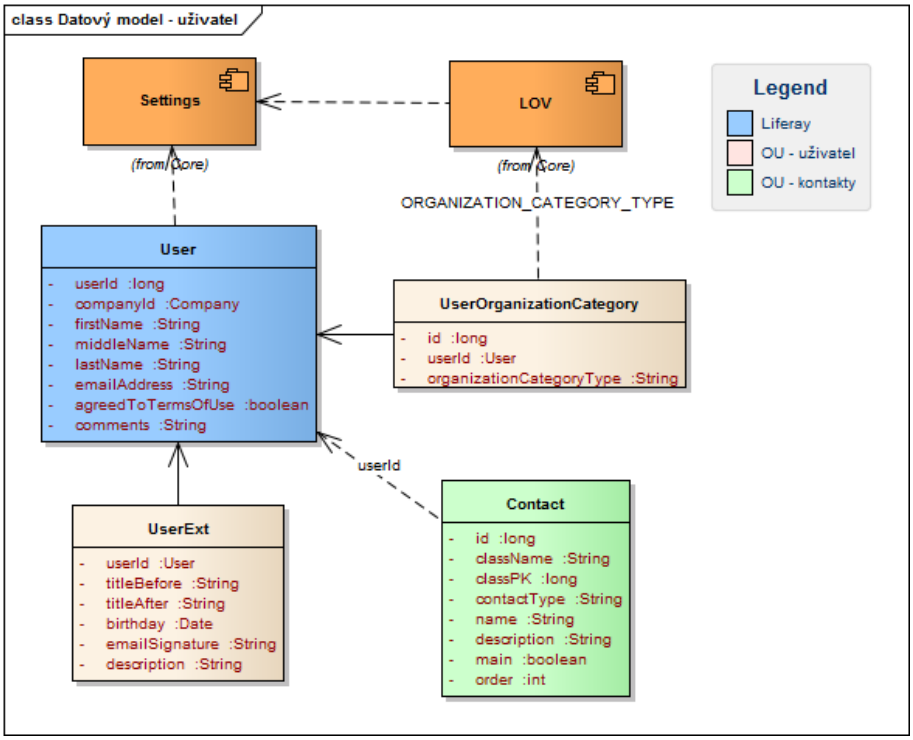
Uživatelé

Uživatel bude mít v portále rozšířené údaje jako např. datum narození nebo tituly před a za jménem. Bude moci vytvořit seznam zájmů na základě kategorie organizace, které chce sledovat, a podle toho se mu budou nabízet např. události organizace, která v oblasti působí. Zájmy budou dostupné v číselnících. Základní funkcionalita uživatelů je znázorněná na obrázku 12.



Obrázek 12: Návrh základní funkcionality uživatelů v portletu OU

Uživatelé se budou registrovat na portál pomoci registračního formuláře určený pro uživatele. Uživatel poté bude moci spravovat svůj profil. Pokud uživatel nebude přiřazen k žádné organizaci, bude na portále vystupovat jako registrovaný uživatel. Portlet bude obsahovat správu uživatelů, které bude spravovat Administrátor. Datový model uživatelů v portletu OU je znázorněn na obrázku 13.

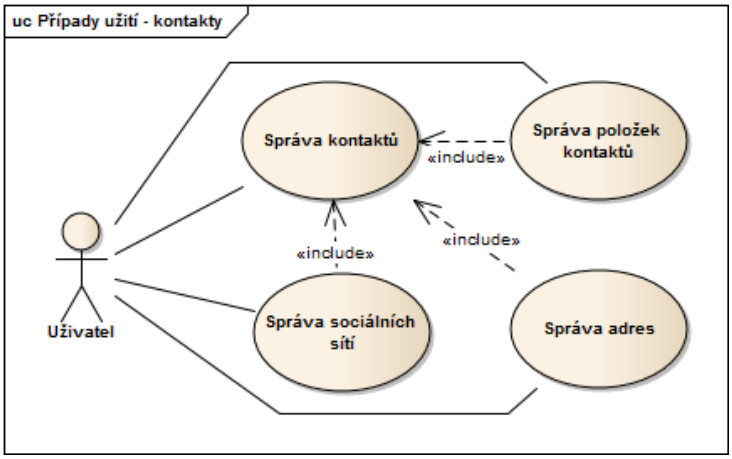


Obrázek 13: Datový model uživatelů v portletu OU

Kontakty

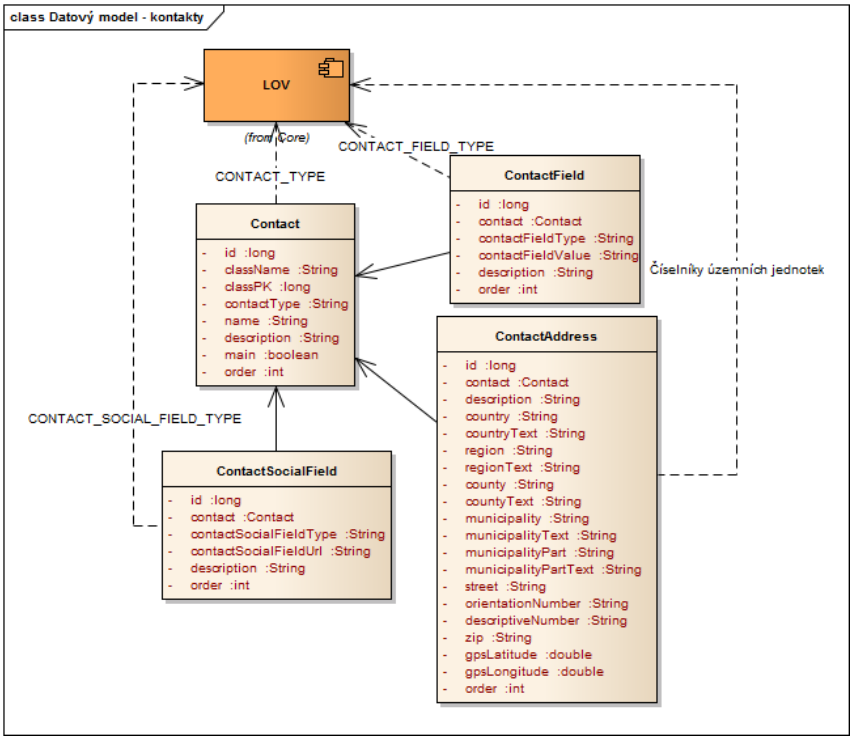
Kontakty budou jednou ze základních částí, které budou hojně využívané na různých místech

v portále. Příkladem mohou být kontakty u organizace, kontakty uživatele, kontaktní osoba u události a jiné. Proto kontakty budou navrženy tak, aby se pokryla velká škála možností zaznamenání kontaktu a umožnit opětovnou použitelnost na více místech. Základní funkcionalita kontaktů je znázorněná na obrázku 14.



Obrázek 14: Návrh základní funkcionality kontaktů v portletu OU

Kontakt bude obsahovat typ kontaktu získaný z číselníku, aby bylo možné kontakt rozpoznat, čeho se týká. Např. může jít o sídlo firmy, trvalé bydliště nebo místo konání události. Bude obsahovat název třídy, ke které bude kontakt navázán, a primární klíč navázané instance třídy.

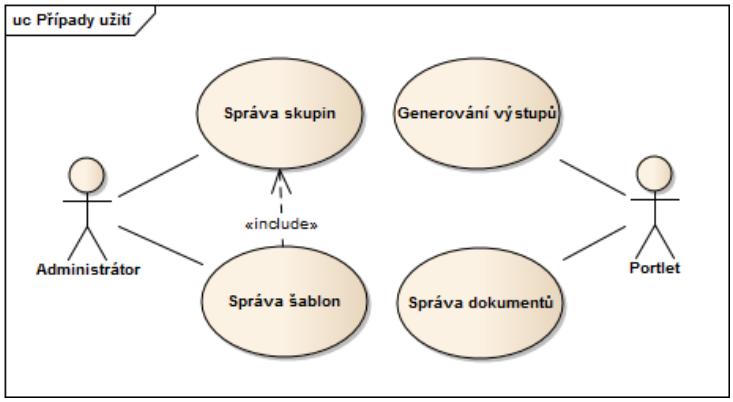


Obrázek 15: Datový model kontaktů v portletu OU

Ke kontaktu může být přidána kompletní adresa zařazená do územních jednotek státu z číselníků (stát, kraj, okres, obec a část obce) včetně GPS souřadnic. Dále ke kontaktu bude možné přidávat položky kontaktu. K položce bude vybrán typ položky z číselníku (telefon, mobil, e-mail, webové stránky a jiné) a bude možné stejný typ přidat vícekrát do jednoho kontaktu např. dvě telefonní čísla. Ke kontaktu bude možné přidávat i odkazy na sociální sítě podle typu sociální sítě, jenž bude vybírán z číselníku. Datový model kontaktů v portletu OU je znázorněn na obrázku 15.

Portlet DocGen

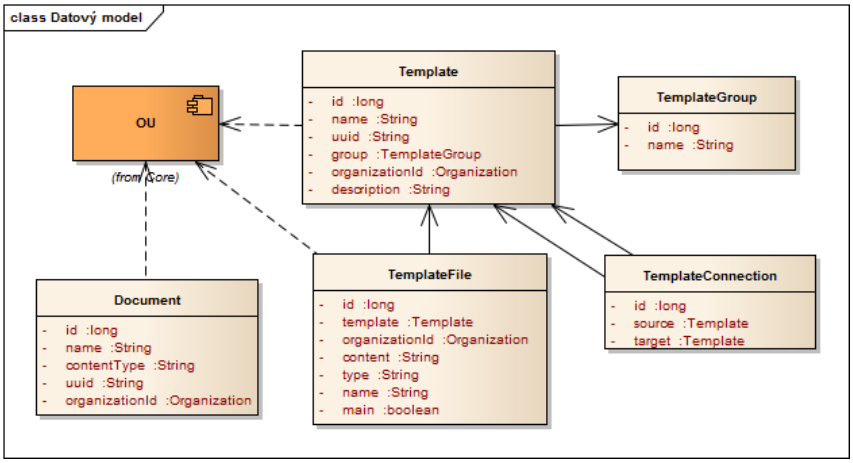
Portlet DocGen bude navržen tak, aby prováděl správu šablon a generoval výstupy pro ostatní portlety. Každá šablona bude jednoznačně identifikovatelná v celém systému pomocí UUID. Šablony budou rozdělené do skupin pro lepší přehlednost a organizaci. Jedna šablona se bude moci skládat z více samostatných souborů. U šablony bude možnost nahrání pozměněného souboru tak, aby se dala definovat u jednotlivé organizace vlastní verze upravené šablony. Jednotlivé šablony budou mezi sebou propojené, aby se eliminovalo opakování definice šablony. Typickým příkladem jsou hlavička a patička, které můžou být pro všechny šablony stejné. Na základě zkušeností, kde zákazníci neumí pracovat se šablonami, bude za šablony odpovídat Administrátor a bude je upravovat podle přání zákazníka. Základní funkcionalita je znázorněná na obrázku 16.



Obrázek 16: Návrh základní funkcionality portletu DocGen

Portlet DocGen nebude připravovat data pro jednotlivé šablony. Šablony budou připravené ve formátu FTL pro Apache FreeMarker nebo ve formátu XML pro DocBook. Zodpovědnost za přípravu dat bude na jednotlivých portletech, které budou volat generování dokumentů. Data se budou připravovat ve formě objektů a budou vkládána do mapy, ze které se budou objekty doplňovat do generovaného výstupu. Portlet DocGen bude mít pouze přímou zodpovědnost za výběr správné šablony či její varianty pro danou organizaci a za propojení šablon, aby se docílilo požadovaného výstupu. Datový model portletu DocGen je znázorněn na obrázku 17.

Portlet DocGen bude také obsahovat rozhraní spravující různé dokumenty a soubory. Bude se jednat např. o dokumenty generované systémem, o dokumenty organizace jako jsou stanovy,

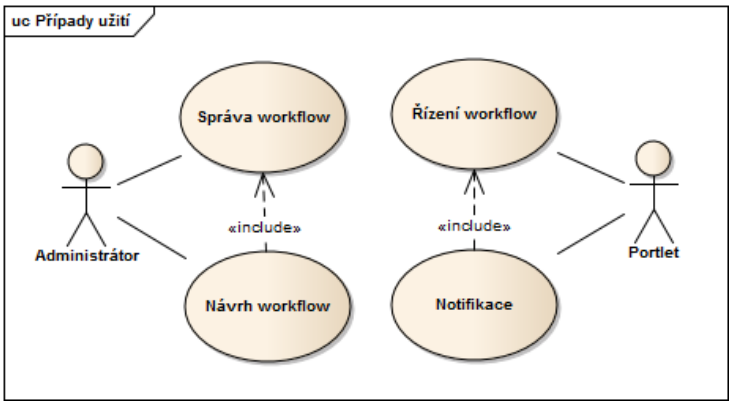


Obrázek 17: Datový model portletu DocGen

přílohy e-mailů a další. Dokumenty se budou ukládat na diskové uložiště tak, aby se minimalizovaly duplicity souborů, kvůli úspoře místa na disku.

Portlet Workflow

Při provozu portálu se předpokládá, že v některých případech bude potřeba usměrnit chování uživatelů. Může se jednat např. o proces schvalování registrace organizace na portále. Portál Liferay poskytuje vlastní jednoduché řešení řízení procesu pomocí workflow, avšak toto řešení není dostačující. Proto bylo rozhodnuto naprogramovat vlastní řešení. Portlet Workflow bude určen k tomu, aby se dal do systému zanést konfigurovatelný proces spustitelný uživatelem a to standardizovaným způsobem. Za správu bude zodpovědný uživatel v roli Administrátor. Základní funkcionalita je znázorněná na obrázku 18.

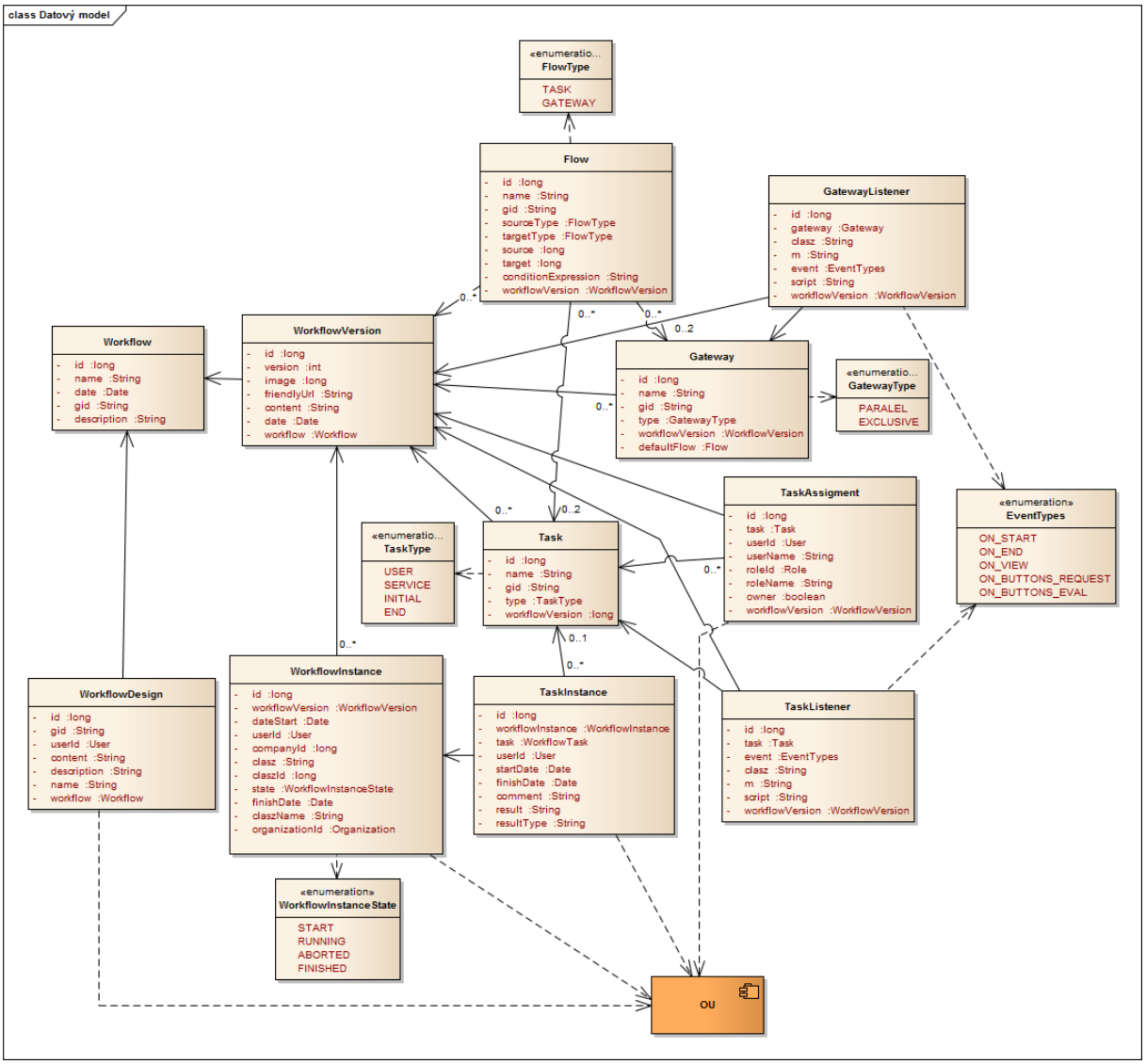


Obrázek 18: Návrh základní funkcionality protletu Workflow

Portlet Workflow bude zodpovědný za správu jednotlivých workflow a bude umožňovat jejich grafickou editaci a spouštění. Jako notace se použije BPMN[33] diagram, který se využije při

[33]http://www.bpmn.org/

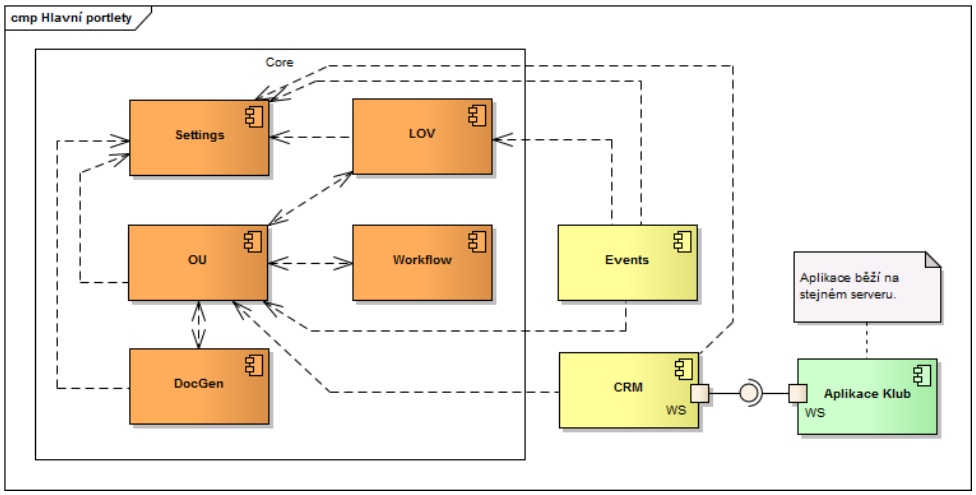
grafické editaci. Aby se dalo workflow spustit, převede se BPMN diagram do datových struktur. Workflow bude mít v rámci systému unikátní UUID, ale bude moci existovat i ve více verzích. Pro nově spuštěné procesy se vždy použije poslední publikovaná verze workflow. BPMN diagram bude možné editovat v grafickém editoru. Vlastnosti, které se budou vykonávat v průběhu procesu, se budou zapisovat do BPMN diagramu pomocí Extension Elements rozšiřující základní diagram. Portlet Workflow bude zodpovědný za notifikaci uživatelů nebo skupin, které mají v procesu vykonat daný krok. Datový model portletu Workflow je znázorněn na obrázku 19.



Obrázek 19: Datový model portletu Workfolow

5.2.2 Hlavní portlety

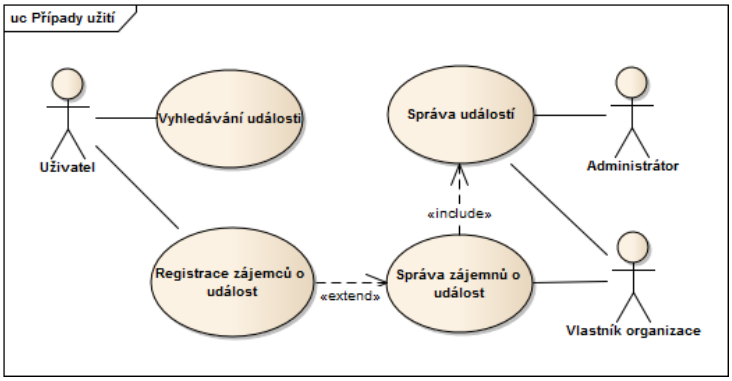
Z hlavních portletů se budou sestavovat konkrétní stránky na portále Pohodlné.info, které se budou zobrazovat uživatelům a mohou je tak využívat pro svou potřebu. Mezi první hlavní portlety nasazené na portál Pohodlné.info budou portlety Events a CRM. Struktura a propojení se základními portlety znázorňuje obrázek 20.



Obrázek 20: Struktura hlavních portletů

Portlet Events

Události na portále budou spravovat registrované organizace. Princip bude vycházet ze známé podstaty událostí, které se objevují v různých aplikacích poskytující kalendář, tzn. může se jednat o celodenní událost, může se opakovat, atd. Základní funkcionalita je znázorněná na obrázku 21.

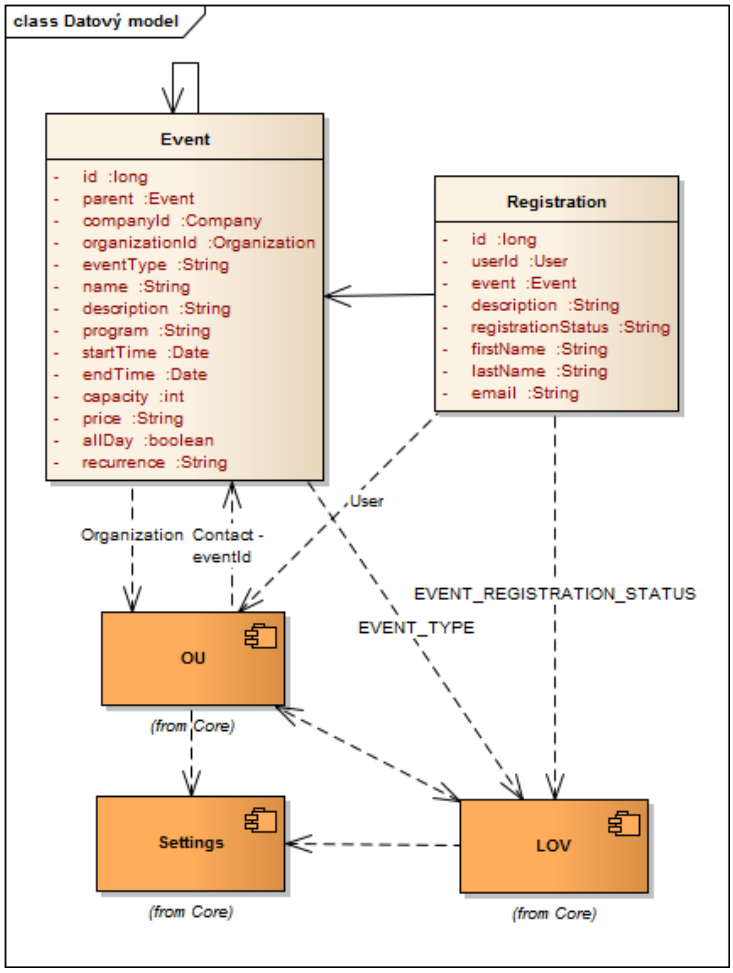


Obrázek 21: Návrh základní funkcionality protletu Events

Událost bude rozšířena o typ události z číselníku podle toho, zda se bude jednat např. o tábor, školní akci, sportovní akci a jiné. Dále bude obsahovat informace o programu, ceně a kapacitě

akce. U události bude uvedeno místo konání a kontaktní osoba, které budou využívat funkčnost kontaktů z portletu OU. K události bude možné přidávat fotky a propagační materiály. V události bude možné přidávat podudálosti, které mohou danou událost rozložit na více časových úseků.

Portál Liferay obsahuje Asset Framework[34], který umožňuje přidávat do vlastních portletů funkční části přímo z portálu Liferay. Toho využijeme při vložení hodnocení a komentářů u události, aniž bychom museli programovat funkčnost navíc. V kódu stačí zaregistrovat do Asset Frameworku objekt události, ke kterému se bude funkční část portálu Liferay vázat.



Obrázek 22: Návrh základní funkcionality protletu Events

K události se bude moci zaregistrovat přihlášený i nepřihlášený uživatel. Ti budou dostávat informace o změnách v událostech. Na druhou stranu organizátor uvidí přehled uživatelů, kteří se o jeho událost zajímají. Datový model portletu Events je znázorněn na obrázku 22.

Události bude možné vyhledávat podle daných kritérií, jako je čas a datum konání, rozsah ceny, místo konání a jiné.

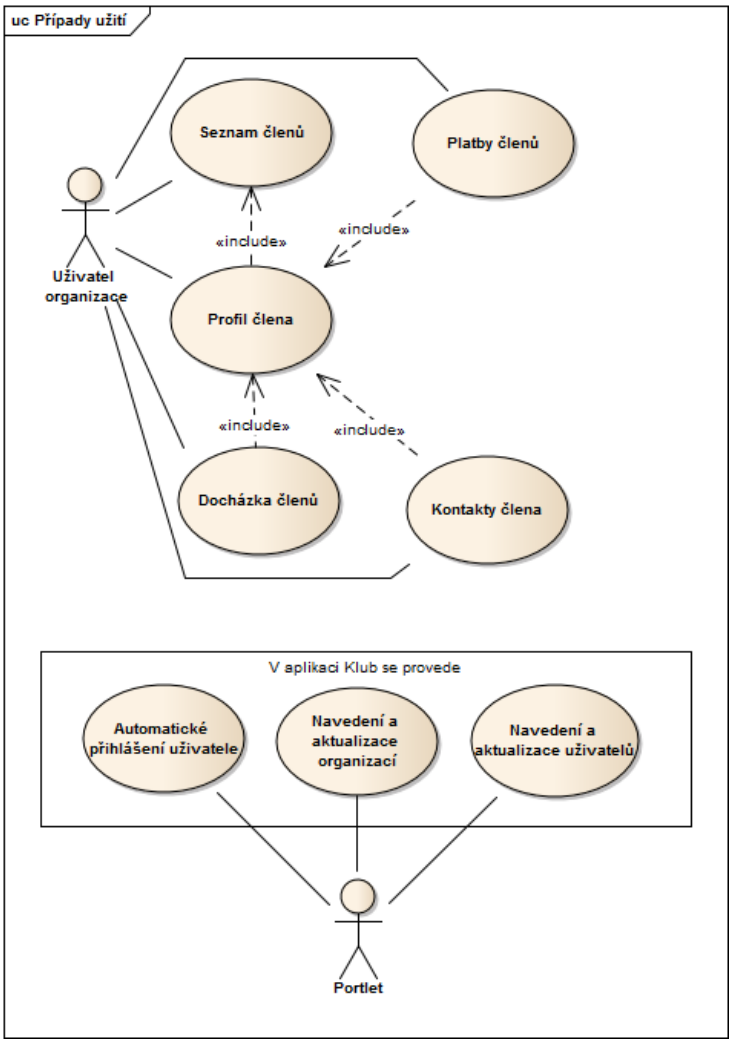
[34]https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/asset-framework

Portlet CRM

Portlet CRM bude vycházet z aplikace Klub. V první fázi dojde k integraci aplikace Klub s tímto portletem, abychom mohli nabídnout zákazníkům aplikaci v novém systému co nejdříve. Portlet CRM bude v této fázi zobrazovat základní údaje o členech přenášené z aplikace Klub pomocí webových služeb.

Správu organizací a uživatelů bude již primárně zajišťovat portál Pohodlné.info a na straně aplikace Klub bude probíhat jen aktualizace údajů o organizaci a uživatelích. Přihlašování a odhlašování do aplikace bude probíhat primárně přes portál Pohodlné.info, na který bude uživatel přesměrován. Rozšířenou správu členů bude muset uživatel provádět přímo v aplikaci Klub, kde bude automaticky přihlášen. Základní funkcionalita portletu je znázorněná na obrázku 23.

V další fázi dojde k analýze převodu aplikace Klub do portletu CRM, která nebude součástí této práce.



Obrázek 23: Návrh základní funkcionality protletu CRM

6 Implementace klíčových komponent určených k integraci

Implementace výše uvedených portletů je realizována pomocí Liferay IDE[35], který je dostupný ke stažení na stránkách portálu Liferay. Pro implementaci portletů je zvolen portál Liferay 6.2 CE s vestavěným aplikačním servrem Apache Tomcat. V Eclipse IDE zvolíme perspektivu pro Liferay, která nastaví prostředí Eclipse pro vývoj portletů. V nastavení prostředí Eclipse je důležité nastavit cesty k samotnému portálu a k aplikačnímu serveru Apache Tomcat, který se bude pouštět skrze prostředí Eclipse v debug módu. Při prvním spuštění portálu dojde k prvotní instalaci, kde je potřeba nastavit název portálu, hlavního administrátora a zvolit typ (MySQL, HSQLDB, Oracle aj.) a název databáze.

Založení nového portletu provedeme pomocí průvodce pro nový projekt New Liferay Plugin Project. V něm vyplníme název a umístění projektu, zvolíme build pomocí Apache Maven a zvolíme typ portletu. Typy protletů jsou:

- **Portlet** - vytvoří samostatnou aplikaci, která poběží jako portlet na portále Liferay.
- **Service Builder** - vytvoří portlet, který bude využívat Service Builder.
- **Hook** - přepisuje nebo rozšiřuje základní funkcionalitu portálu Liferay.
- **Layout Template** - umožňuje přidat nové vlastní rozložení stránek v portále Liferay.
- **Theme** - definuje vlastní vzhled portálu Liferay.
- **Ext** - umožňuje přepsat některé třídy používané portálem Liferay
- **Web** - webový modul, který umožňuje integraci portálu Liferay s jinými systémy.

Pro implementované portlety jsme použili typ Service Builder[36], protože potřebujeme pro portlet nadefinovat datovou strukturu a vygenerovat pro portlet rozhraní pro komunikaci s ostatními portlety. Projekt bude rozdělen na dva podprojekty. V prvním podprojektu se implementuje funkcionalita portletu, který bude sestaven jako webová aplikace a bude nasazen na portál Liferay. Do druhého podprojektu se generuje rozhraní, které bude vystavené a dostupné pro ostatní portlety a bude sestaven jako samostatná knihovna. Knihovna může být buď součástí jen webové aplikace nebo umístěna mezi knihovny aplikačního serveru Apache Tomcat a může být takto dostupná pro ostatní webové aplikace. Datová struktura se definuje v souboru `service.xml`, kde pro každou entitu bude existovat rozhraní jako lokální služba uvnitř portálu Liferay nebo jako webová služba.

V dalším kroku průvodce vybereme typ frameworku JSF 2.x pro vývoj portletů v JSF a dáme dokončit. Poté se v Eclipse IDE vytvoří nový projekt.

[35] https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/liferay-ide
[36] https://dev.liferay.com/develop/tutorials/-/knowledge_base/6-2/service-builder

Vyvíjené portlety lze v Liferay IDE rovnou navázat na portál Liferay. Po spuštění portálu Liferay v debug módu se portlety automaticky nainstalují do portálu Liferay a lze takto testovat jejích funkčnost.

V rámci vývoje jsem dostal za úkol implementovat portlety Settings a DocGen. Ostatní portlety dostali za úkol implementovat další zaměstnanci a nejsou součástí této práce.

6.1 Implementace portletu Settings

Pro portlet Settings byl vytvořen projekt `pi-settings` typu Service Builder, který obsahuje podprojekty `pi-settings-portlet` a `pi-settings-portlet-service`. Do `service.xml` v projektu `pi-settings-portlet` se zanesly tři entity z obrázku 7, včetně atributů. Byla přidána jedna entita `SettingsUtils` bez atributů, která vytvoří pouze lokální službu, která zajišťuje funkcionalitu z obrázku 6 na úrovni aplikační vrstvy. Prezentační vrstva byla sestavena pomocí JSF a obsahuje Správu skupin, Správu klíčů a Správu hodnot. Funkcionalitu grafických prvků s aplikační vrstvou budou zajišťovat třídy Java Beans. Ukázky grafických částí portletu jsou na obrázcích 29 a 30 (Příloha C).

6.2 Implementace portletu DocGen

Pro portlet DocGen byl vytvořen projekt `pi-docgen` typu Service Builder, který obsahuje podprojekty `pi-docgen-portlet` a `pi-docgen-potlet-service`. Do `pom.xml` projektu `pi-docgen-portlet` byly přidány závislosti na portlety Settings a OU, závislosti na knihovny Apache FreeMarker a DocBook. Do `service.xml` v projektu `pi-docgen-portlet` se zanesly čtyři entity z obrázku 17, včetně atributů. Byly přidány dvě entity `DocgenUtils` a `DocumentUtils` bez atributů, které vytvoří pouze lokální služby, které zajišťují funkcionalitu z obrázku 16 na úrovni aplikační vrstvy. Prezentační vrstva byla sestavena pomocí JSF a obsahuje Správu skupin a Správu šablon. Funkcionalitu grafických prvků s aplikační vrstvou budou zajišťovat třídy Java Beans. Ukázky grafických částí portletu jsou na obrázcích 31, 32 a 33 (Příloha D).

7 Návrh postupu nasazení optimalizovaných produktů u zákazníka

Portál Pohodlné.info bude zastřešovat tři oblasti, které jsou mezi sebou provázané. Jedná se o školství, sport a dětské tábory. Na základě toho bude portál Pohodlné.info rozdělen na několik tématických částí:

- **České-školy.info** budou vycházet ze současného portálu České-školy.info, budou obsahovat informace z českého školství a nabídku produktů firmy z této oblasti.
- **Sportovní-kluby.info** budou obsahovat informace z oblasti sportu a nabídku produktů firmy určené pro sportovní kluby.
- **Dětské-tábory.info** budou vycházet ze současné aplikace Katalog táborů, budou obsahovat informace kolem táborů a nabídku produktů firmy určené pro organizátory táborů.
- **Interní firemní stránky** budou složit k interním potřebám firmy a bude se převážně jednat o firemní CRM.

Na základě skutečnosti, že Česká školní inspekce vyvinula systém InspIS (Inspekční informační systém), který zcela zdarma nabízí školám jako nástroj pro tvorbu ŠVP podobný firemní aplikaci Smile, začalo docházet k úbytku zákazníků. Proto bylo nakonec rozhodnuto, že aplikace Smile nebude převedena do nového prostředí a bude pouze udržována pro stávající zákazníky.

Pro aplikaci ISIS bude vyčleněna samostatná etapa, protože bude potřeba připravit nový portál postavený na portále Liferay. Současná aplikace přináší komplikace v podobě nefunkčnosti po instalaci poslední verze Javy a aplikačního serveru Apache Tomcat, což přináší bezpečností rizika na straně zákazníka. Proto není jiná možnost, než tuto aplikaci kompletně přepsat pro portál Liferay.

7.1 Etapy nasazení

Na základě nasbíraných poznatků z předešlých kapitol je patrné, že všechny aplikace nelze přepsat do nového prostředí najednou. Proto byly naplánovány etapy tak, aby se to nedotklo hned všech zákazníků a k přechodu docházelo postupně. Byl sestaven plán jednotlivých prací uvedený níže. Odhady vychází z předpokladu, že pro úkoly bude vyčleněn jeden měsíční pracovní úvazek. Počet úvazků může být navýšen, pokud bude potřeba urychlit řešení některých úkolů.

V průběhu psaní diplomové práce se již některé kroky začaly realizovat a Etapa I. je již splněná a uvedena do provozu.

Etapa I.

Odhadovaná doba realizace: **Říjen 2015 - Březen 2016**

1. **Instalace a konfigurace portálu Pohodlné.info (1 měsíc).**
    - Instalace portálu Liferay na produkčním serveru.
    - Konfigurace portálu pro Pohodlné.info.
    - Uvolnění portálu Pohodlné.info do ostrého provozu.
2. **Vývoj a nasazení základních portletů (4 měsíce).**
    - Implementace a testování základního portletů Settings.
    - Implementace a testování základního portletu LOV.
    - Implementace a testování základního portletu OU.
    - Implementace a testování základního portletu DocGen.
    - Implementace a testování základního potletu Workflow.
    - Nasazení základních portletů na portál Pohodlné.info.
    - Konfigurace administrátorských stránek využívající základní portlety.
    - Příprava grafického designu.
    - Nasazení grafického designu.
3. **Přesun a optimalizace portálu České-školy.info (1 měsíc).**
    - Migrace institucí a uživatelů z portálu České-školy.info na portál Pohodlné.info.
    - Aktualizace a nasazení portletu ŠVP.
    - Aktualizace a nasazení portletu Studijní materiály.
    - Příprava stránek na Pohodlné.info pro novou část České-školy.info.
    - Konfigurace stránek pro správu ŠVP.
    - Konfigurace stránek pro správu Studijních materiálů.
    - Testování propojení aplikace Smile s portálem Pohodlné.info.
    - Uvolnění části České-školy.info do ostrého provozu.

Etapa II.

Odhadovaná doba realizace: **Duben 2016 - Srpen 2016**

1. **Vývoj a nasazení hlavního portletu CRM (3 měsíce).**
    - Implementace a testování hlavního portletu CRM.
    - Implementace rozhraní pro integraci portletu CRM a aplikace Klub.
    - Úprava integrovatelných částí na straně aplikace Klub.

- Testování integrace portletu CRM s aplikací Klub.
- Nasazení portletu CRM na portál Pohodlné.info.

2. **Vývoj a nasazení hlavního portletu Events (1 měsíc).**
   - Implementace a testování portletu Events.
   - Nasazení portletu Events.
3. **Vytvoření části Sportovní-kluby.info (1 měsíc).**
   - Příprava stránek na Pohodlné.info pro novou část Sportovní-kluby.info.
   - Konfigurace stránky pro katalog sportovních klubů.
   - Konfigurace stránky pro vyhledávání sportovních událostí a zobrazení výsledků vyhledávání.
   - Konfigurace stránek pro evidenci členské základny.
   - Konfigurace a zprovoznění zasílání obchodních sdělení.
   - Uvolnění části Sportovní-kluby.info do ostrého provozu.

Etapa III.

Odhadovaná doba realizace: **Září 2016 - Březen 2017**

1. **Aktualizace hlavního portletu CRM (1 měsíc).**
   - Implementace a testování rozšířených částí portletu CRM.
   - Nasazení nové verze portletu CRM.
2. **Vytvoření interní firemní části (1 měsíc).**
   - Příprava stránek na Pohodlné.info pro interní firemní část.
   - Vytvoření přístupů pro zaměstnance firmy.
   - Konfigurace stránek pro interní CRM.
   - Uvolnění interní části do ostrého provozu.
3. **Migrace dat ze současných dvou verzí CRM (2 měsíce).**
   - Příprava dat pro migraci databáze z aplikace CRM v1.
   - Migrace dat z aplikace CRM v1 do databáze na portále Pohodlné.info.
   - Přesun uživatelských souborů z aplikace CRM v1.
   - Odstavení aplikace CRM v1.
   - Příprava dat pro migraci databáze z aplikace CRM v2.

- Migrace dat z aplikace CRM v2 do databáze na portále Pohodlné.info.
- Přesun uživatelských souborů z aplikace CRM v2.
- Odstavení aplikace CRM v2.

4. **Úprava části Sportovní-kluby.info (1 měsíc).**
   - Konfigurace stránek pro evidenci členské základny pro nové CRM.
5. **Migrace dat z aplikace Klub (2 měsíce).**
   - Příprava dat pro migraci databáze z aplikace Klub.
   - Migrace dat z aplikace Klub do databáze na portále Pohodlné.info.
   - Přesun uživatelských souborů z aplikace Klub.
   - Odstavení aplikace Klub a přesměrování na nové stránky.

Etapa IV.

Odhadovaná doba realizace: **Duben 2017 - Srpen 2017**

1. **Aktualizace hlavního portletu Events (1 měsíc).**
   - Implementace a testování rozšířených částí portletu Events.
   - Nasazení nové verze portletu Events.
2. **Úprava částí Sportovní-kluby.info a České-školy.info (1 měsíc).**
   - Konfigurace stránek s událostmi na novou verzi.
   - Konfigurace stránky pro vyhledávání školních událostí a zobrazení výsledků vyhledávání.
3. **Vytvoření části Dětské-tábory.info (1 měsíc).**
   - Příprava stránek na Pohodlné.info pro novou část Dětské-tábory.info.
   - Konfigurace stránky pro katalog táborů.
   - Konfigurace stránky pro vyhledávání táborů zobrazení výsledků vyhledávání.
   - Uvolnění části Dětské-tábory.info do ostrého provozu.
4. **Migrace dat z apikace Katalog táborů (2 měsíce).**
   - Příprava dat pro migraci databáze z aplikace Katalog táborů.
   - Migrace dat z aplikace Katalog táborů do databáze na portále Pohodlné.info.
   - Přesun uživatelských souborů z aplikace Katalog táborů.
   - Odstavení apliakce Katalog táborů a přesměrování na nové stránky.

Etapa V.

Jedná se o prvotní odhad prací, bude upřesněno v průběhu následujících měsíců.
Odhadovaná doba realizace: **Září 2017 - Červen 2018**

1. **Příprava instalace nového portálu ISIS (1 měsíc).**
   - Instalace portálu Liferay na testovacím serveru.
   - Konfigurace portálu pro ISIS pro čistou instalaci bez testovacích dat.
   - Nasazení základních portletů.
2. **Vytvoření nových portletů pro ISIS (4 měsíce).**
3. **Vytvoření stránek na portále ISIS (1 měsíc).**
4. **Nasazení portálu ISIS u zákazníka (1 měsíc).**
   - Vytvoření kopie instalace portálu ISIS včetně databáze.
   - Přenesení kopie portálu ISIS na server zákazníka.
   - Konfigurace portálu ISIS u zákazníka podle požadavků.
5. **Migrace dat ze staré verze do nové (3 měsíce).**
   - Příprava dat pro migraci databáze z původní aplikace ISIS.
   - Migrace dat z aplikace ISIS do databáze nového portálu ISIS.
   - Přesun uživatelských souborů mezi systémy.
   - Odstavení aplikace současné verze portálu ISIS a přesměrování na nový portál.

8 Závěr

Tato práce shrnula způsob, jak probíhá inovace produktů a nastavení vnitřních procesů ve firmě VRK plus s.r.o.. Práce obsahuje popis slučovaných produktů firmy a popis technologií, na kterých byly tyto produkty postaveny. Součástí práce je i návrh optimalizovaných procesních postupů pomocí agilní metody FDD, včetně procesu vyřizování přidělených úkolů. Obsahuje popis nových vývojových nástrojů, které se budou ve firmě primárně používat při vývoji aplikací pro portál Liferay. Byla provedena analýza společných integrovatelných částí v současných produktech, které budou implementovány formou portletů. Byl navržen proces integrace vývoje, jak bude probíhat vývoj jednotlivých portletů. Práce popisuje způsoby i návrh integrace se stávajícími produkty. Součástí je specifikace základních a hlavních portletů, které budou na sobě závislé. Neveřejná část práce obsahuje zdrojové kódy portletů Settings a DocGen, které jsem v rámci této práce vytvořil. Byl navržen časový harmonogram prací pro jednotlivé etapy.

V současné době se projekt nachází v Etapě II. vývoje. Portál Pohodlné.info je již nainstalován a nakonfigurován na produkčním serveru. Jsou nasazeny všechny základní portlety a část České-školy.info běží v ostrém provozu. Nyní probíhá vývoj hlavního portletu CRM. Předpokládá se, že implementace bude probíhat v rámci navrženého časového harmonogramu.

Portál Liferay je z pohledu firmy zajímavý produkt kvůli toho, že jej využívají i velké korporátní firmy. Tudíž je motivací tvůrců jej dále vyvíjet a udržovat jej podle nejnovějších standardů. Znalost portálu Liferay přináší firmě možnost se účastnit projektů jako dodavatel vyvíjející komponenty pro další firmy, které mají zájem využívat portál Liferay pro své podnikání.

Literatura

[1] BECK, Kent at al. *Manifesto for Agile Software Development*, http://www.agilealliance.org.

[2] BUCHALCEVOVÁ, Alena. *Metodiky vývoje a údržby informačních systémů: kategorizace, agilní metodiky, vzory pro návrh metodiky.* 1. vyd. Praha: Grada, 2005. Management v informační společnosti. ISBN 80-247-1075-7.

[3] COAD, Peter, Eric LEFEBVRE a Jeff DE LUCA. *Java modeling in color with UML: enterprise components and process.* Upper Saddle River: Prentice Hall, c1999. ISBN 0-13-011510-X.

[4] PALMER, Stephen R a John M FELSING. *A practical guide to feature-driven development.* Upper Saddle River: Prentice Hall PTR, c2002. Coad series. ISBN 0-13-067615-2.

[5] HOHPE, Gregor a Bobby WOOLF. *Enterprise integration patterns: designing, building, and deploying messaging solutions.* San Francisco: Addison-Wesley, c2004. Addison-Wesley signature series. ISBN 0-321-20068-3.

Dokumentace firmy VRK plus s.r.o.

[6] VRK plus s.r.o, *Technická dokumentace aplikace Smile*

[7] VRK plus s.r.o, *Dokumentace uživatele aplikace Smile*, http://www.vrk.cz/help/user/

[8] VRK plus s.r.o, *Dokumentace správce aplikace Smile*, http://www.vrk.cz/help/admin/

[9] VRK plus s.r.o, *Technická dokumentace aplikace ISIS*

[10] VRK plus s.r.o, *Uživatelská dokumentace aplikace ISIS*, http://smile.vrk.cz:8000/help_isis/index.jsp

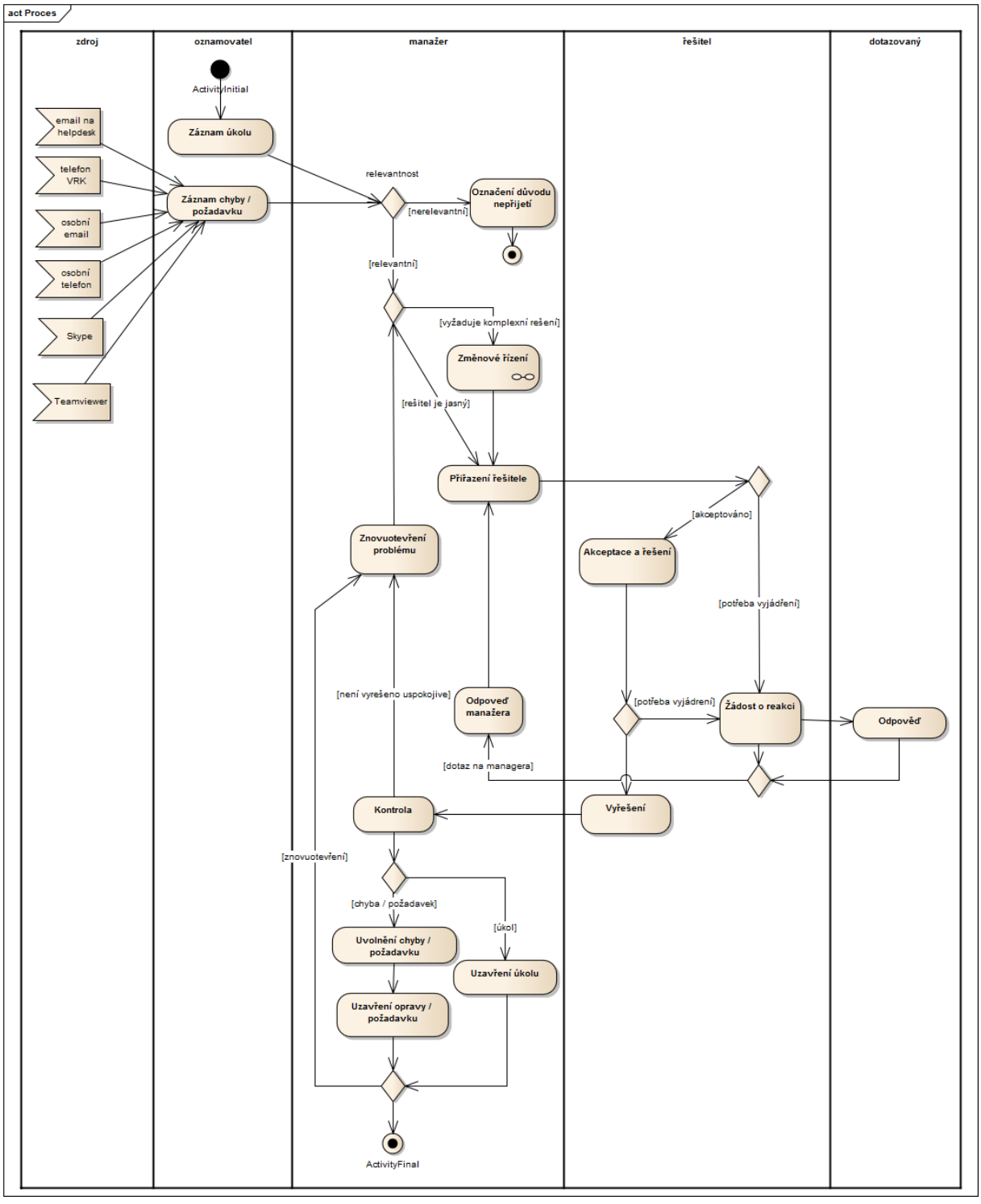
[11] VRK plus s.r.o, *Technická dokumentace aplikace Klub*

[12] VRK plus s.r.o, *Uživatelská dokumentace aplikace Klub*, http://smile.vrk.cz:8000/help_gk/index.jsp

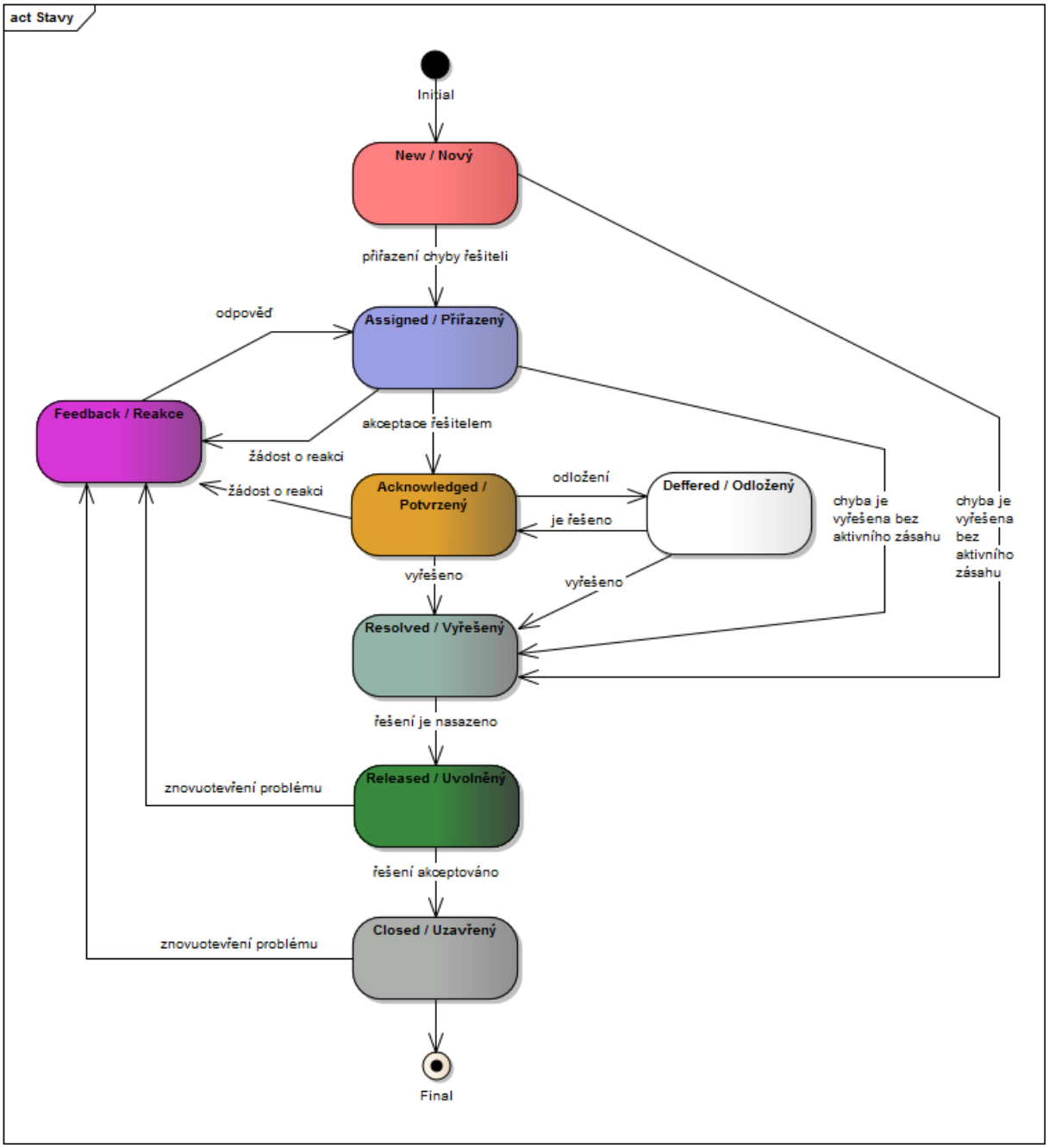
[13] VRK plus s.r.o, *Technická dokumentace aplikace CRM*

[14] VRK plus s.r.o, *Technická dokumentace portálu České-školy.info*

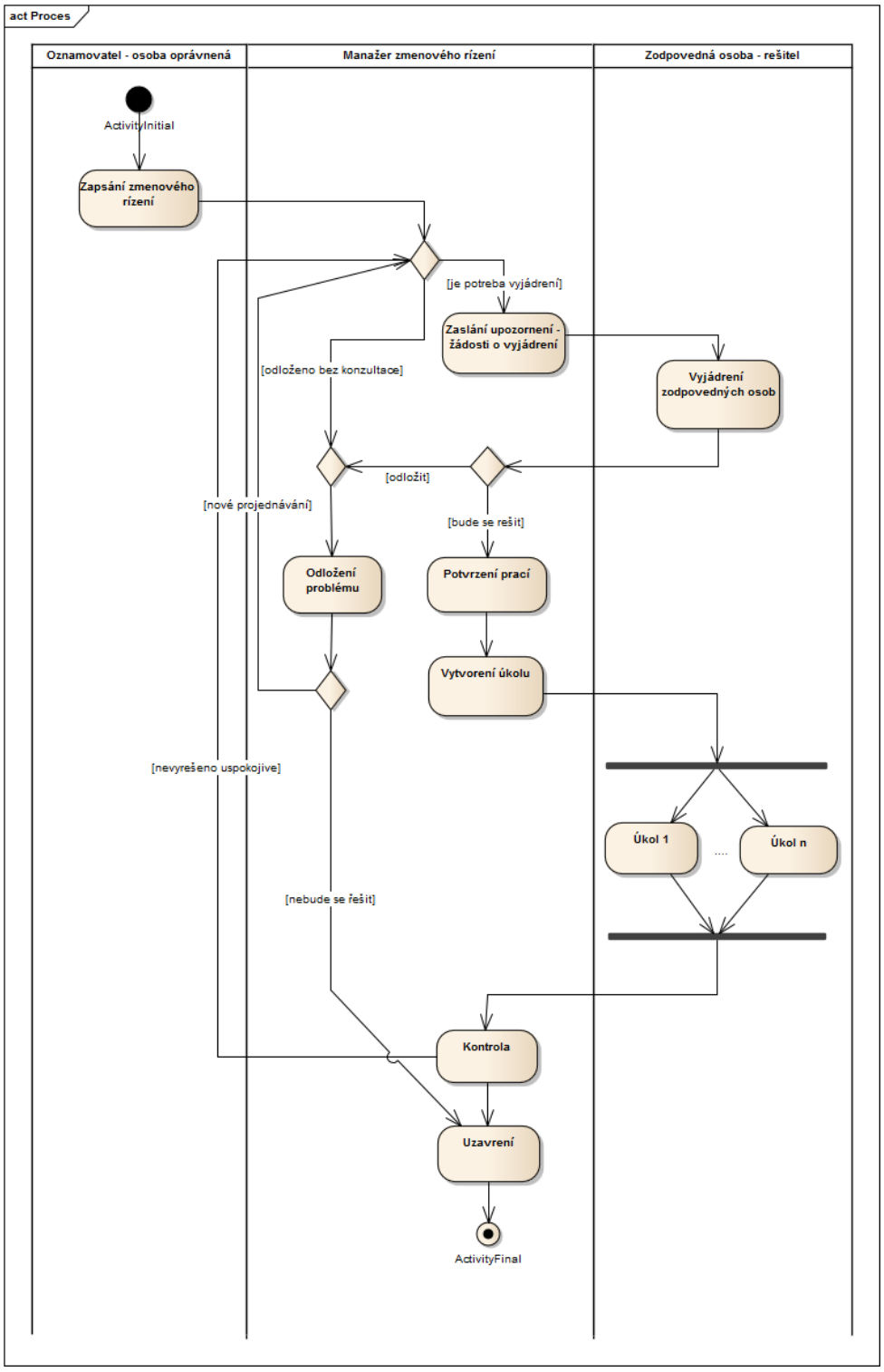
A Procesy a stavy řešení požadavků



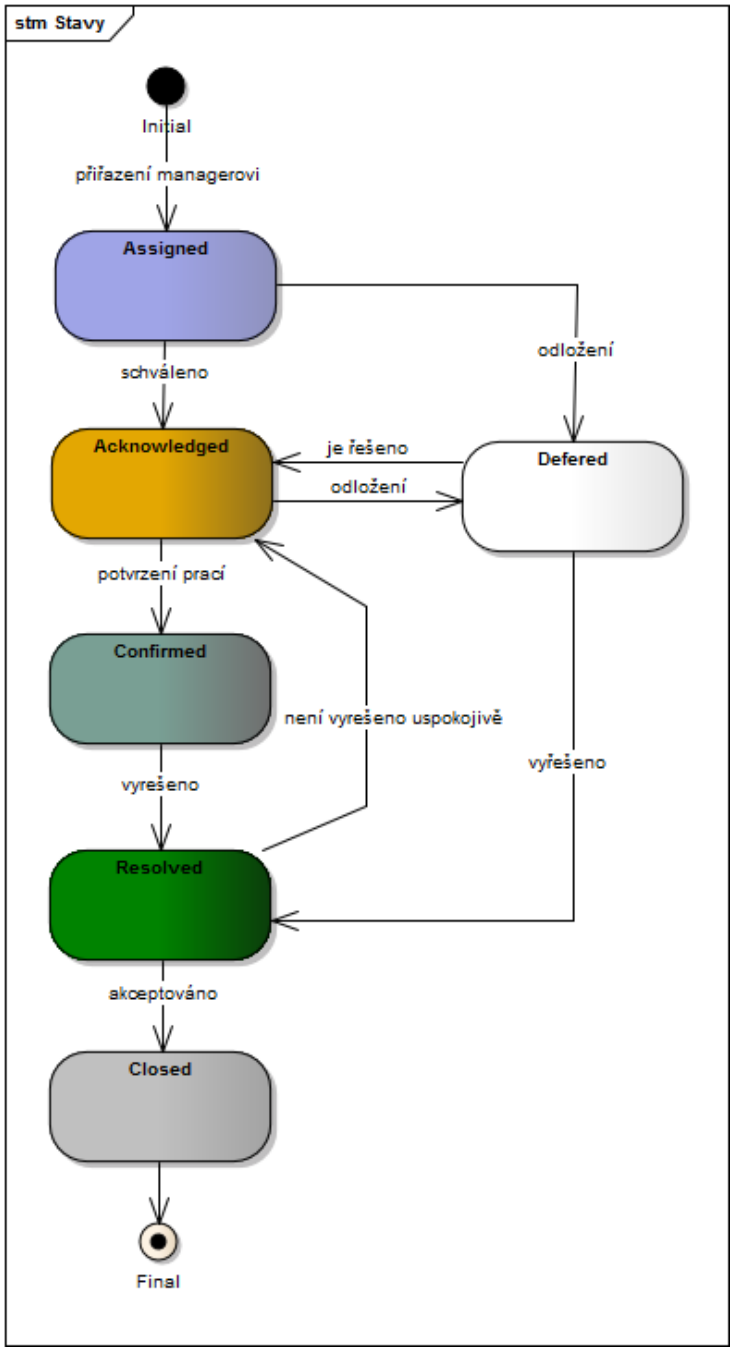
Obrázek 24: Proces řešení požadavků a úkolů



Obrázek 25: Stavový diagram řešení požadavků

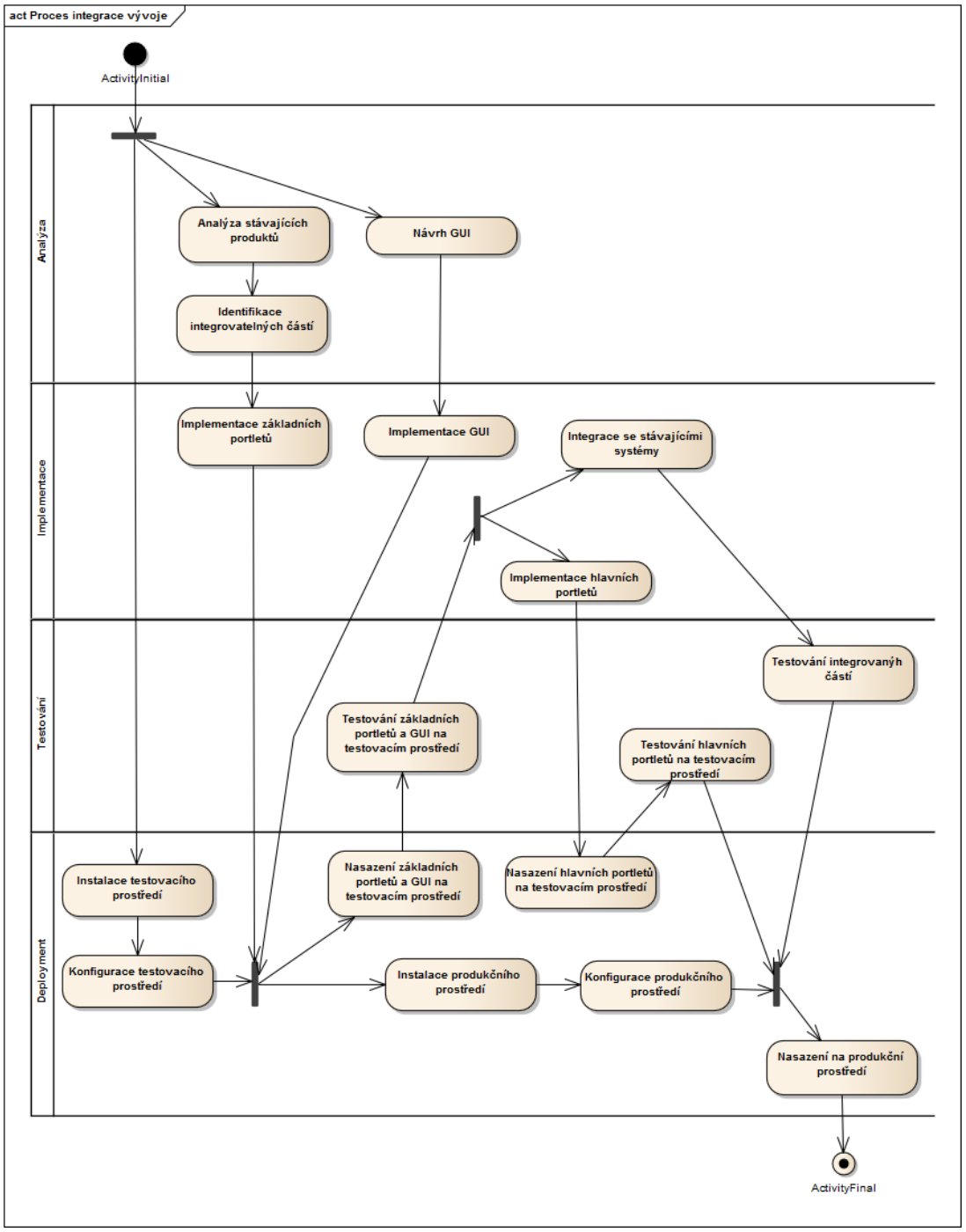


Obrázek 26: Proces změnového řízení



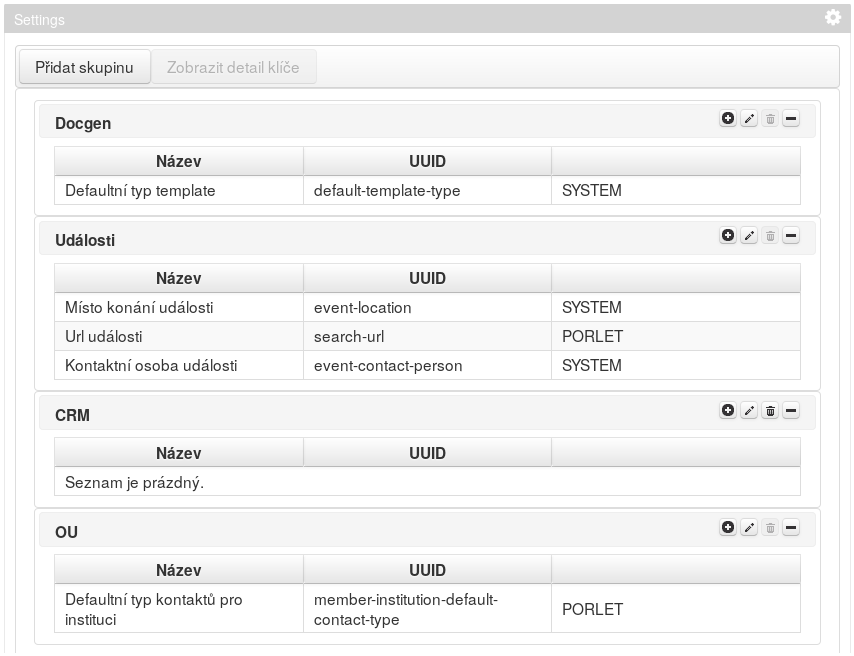
Obrázek 27: Stavový diagram změnového řízení

B Proces integrace vývoje produktů

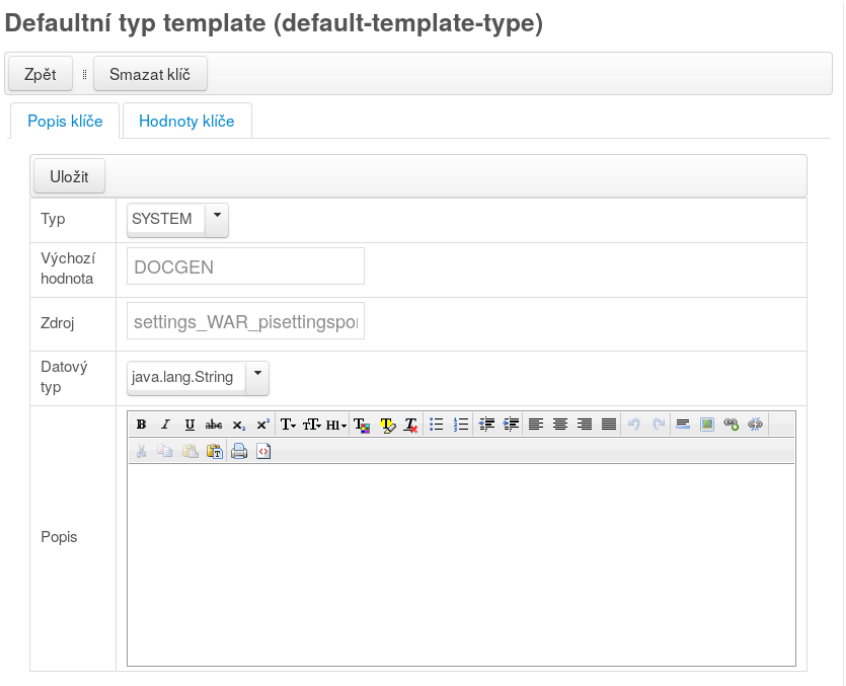


Obrázek 28: Proces integrace vývoje produktů

C Ukázka portletu Settings

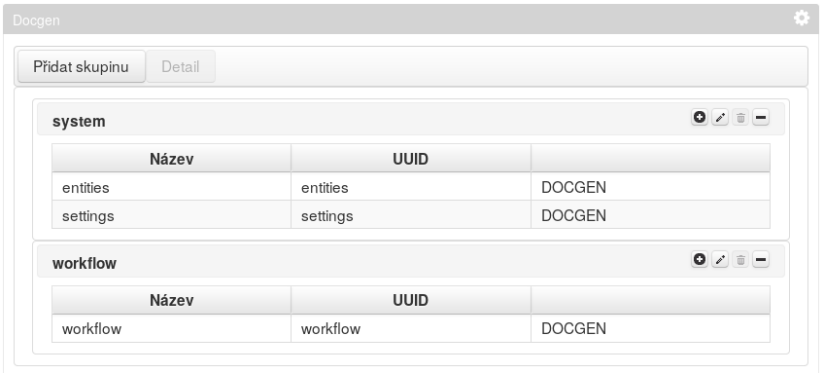


Obrázek 29: Seznam skupin a klíčů nastavení

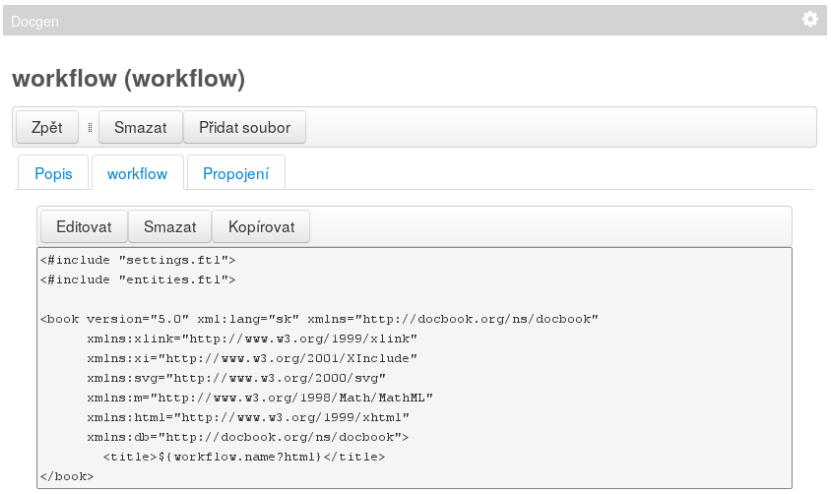


Obrázek 30: Detail klíče nastavení

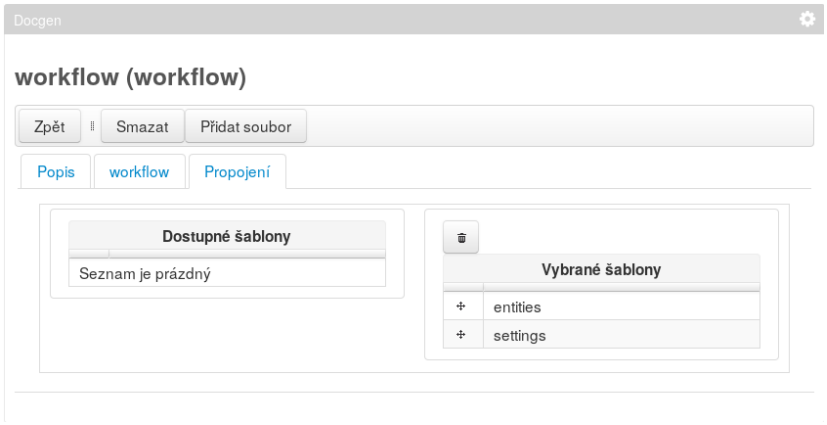
D Ukázka portletu DocGen



Obrázek 31: Seznam skupin a šablon



Obrázek 32: Detail šablony



Obrázek 33: Propojení šablon